

Authenticated Outlier Mining for Outsourced Databases

Boxiang Dong , Hui Wang, Anna Monreale , Dino Pedreschi, Fosca Giannotti, and Wenge Guo

Abstract—The Data-Mining-as-a-Service (DMaS) paradigm is becoming the focus of research, as it allows the data owner (client) who lacks expertise and/or computational resources to outsource their data and mining needs to a third-party service provider (server). Outsourcing, however, raises some issues about *result integrity*: how could the client verify the mining results returned by the server are both sound and complete? In this paper, we focus on outlier mining, an important mining task. Previous verification techniques use an authenticated data structure (ADS) for correctness authentication, which may incur much space and communication cost. In this paper, we propose a novel solution that returns a probabilistic result integrity guarantee with much cheaper verification cost. The key idea is to insert a set of artificial records (*ARs*) into the dataset, from which it constructs a set of artificial outliers (*AOs*) and artificial non-outliers (*ANOs*). The *AOs* and *ANOs* are used by the client to detect any incomplete and/or incorrect mining results with a probabilistic guarantee. The main challenge that we address is how to construct *ARs* so that they do not change the (non-)outlierness of original records, while guaranteeing that the client can identify *ANOs* and *AOs* without executing mining. Furthermore, we build a strategic game and show that a Nash equilibrium exists only when the server returns correct outliers. Our implementation and experiments demonstrate that our verification solution is efficient and lightweight.

Index Terms—Authentication, outsourcing, outlier mining, probabilistic guarantees, game theory

1 INTRODUCTION

THE ability to generate and collect massive amounts of data has grown exponentially in the past years. This rises new challenges in data analytics to extract valuable insights. Fortunately, the advent in networking technologies make it possible to transmit large volume of data through the Internet. It allows data owners, especially those who have large volume of data but limited budget for data analysis, to outsource their data and data mining needs to a third-party service provider. This is referred as the *Data-Mining-as-a-Service (DMaS)* model [42], [49], which provides a cost-effective computing infrastructure for those users who have limited resources for sophisticated data analytics. The model allows the data owner to leverage hardware and software solutions provided by *DMaS* providers, without developing their own.

In this paper, we focus on *outlier mining*, which is to find data objects that do not comply with the general patterns of the majority. Outlier detection plays a critical role in many real-world applications such as computer system intrusion detection, credit card fraud detection and industrial damage

detection. The problem of outlier detection has been widely studied in the data mining community [2], [6], [21], [43]. Previous work [2], [21] show that outlier detection is of high computational complexity; it can be prohibitive if the data has high dimensionality. Although the researchers have proposed several optimization approaches [4], [43] to improve the efficiency of outlier detection, it is difficult for the data owner who lacks the expertise to exploit these techniques. Outsourcing outlier mining to a *DMaS* service provider becomes a natural solution.

Although the *DMaS* paradigm is beneficial for the data owner (client) with limited capabilities to perform sophisticated analysis on their large volume of data, it brings several security challenges. One major concern is the *integrity* of the mining results returned by the service provider (server). Given the fact that the server is potentially untrusted, it is necessary for the client to authenticate if the outliers returned by the server are correct. There are many reasons for the server to return wrong results. For example, the server may return wrong mining results accidentally due to software bugs; it may keep part of the mining results to itself intentionally so that it can sell the retained results to the competitors of the client for profit. There also exists a strong financial incentive for the server to reduce the computational cost. For example, the server may execute the outlier mining on a portion of the outsourced dataset, and charge the client for mining of the whole dataset. The primary challenge in authenticating the outsourced outlier mining arises from the weak computational resources at the client side.

Following [3], we consider three types of the servers that return incorrect mining results: (1) the *Class I* server that is not aware of the authentication process by the client; (2) the

- B. Dong is with the Department of Computer Science, Montclair State University, Montclair, NJ 07043. E-mail: dongb@montclair.edu.
- H. Wang is with the Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ 07030. E-mail: Hui.Wang@stevens.edu.
- A. Monreale and D. Pedreschi are with the Computer Science Department, University of Pisa, Pisa, PI 56126, Italy. E-mail: {annam, pedre}@di.unipi.it.
- F. Giannotti is with ISTI-CNR, Pisa 56127, Italy. E-mail: fosca.giannotti@isti.cnr.it.
- W. Guo is with the Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, NJ 07102. E-mail: wenge.guo@njit.edu.

Manuscript received 17 Apr. 2017; revised 12 Sept. 2017; accepted 16 Sept. 2017. Date of publication 21 Sept. 2017; date of current version 18 Mar. 2020. (Corresponding author: Boxiang Dong.)
Digital Object Identifier no. 10.1109/TDSC.2017.2754493

Class II server that is aware of the fact that the client may perform authentication on mining result, but does not have details of the authentication process; and (3) the *Class III* server that knows the details of the authentication process and tries to avoid being caught by verification. The goal of our verification method is to check if the outliers returned by any of these three types of servers are both *sound* and *complete*. By *soundness*, we mean that each returned tuple is a true outlier. By *completeness*, we mean that all true outliers must be returned by the server.

In consistence with the other work [11], [50], we do not require a *trusted* third-party that performs the authentication for the client. A seemingly straightforward solution is that for each returned record, the client checks if it is indeed an outlier against the original dataset. Though simple, this approach requires computations of $O(nk)$ complexity, where n is the size of the dataset and k is the number of outliers returned by the server. Due to the large volume of data, this naive approach may not be feasible, especially if the client uses a resource-constrained device (e.g., a smart phone) that has weak computational power for authentication. Moreover, this solution cannot verify the *completeness* of the outliers.

In theory, the techniques that verify general-purpose computation [5], [19] can be applied to authenticate any outsourced computation task. These verification techniques require a preprocessing phase in which the client generates auxiliary information of the outsourced computation. Then the server constructs interactive proofs or probabilistically checkable proofs (PCPs) to demonstrate the correctness of the returned result. Unfortunately, this body of theory is considered to be impractical, due to the complexity of the preprocessing step and the expensive cost of using general-purpose cryptographic proofs for data mining problems. A large body of the existing authentication techniques use an authenticated data structure (ADS) (e.g., [24], [52]). The key idea is that the client constructs an ADS from her dataset, and sends both the dataset and ADS to the server. During query processing, the server traverses the ADS and constructs a verification object (VO) that proves the correctness of the results. The VO is sent back to the client together with the query results for verification. Though effective, ADS-based authentication may incur significant space and communication cost.

We propose an efficient *probabilistic* authentication framework for outsourced outlier mining. The architecture of our approach is illustrated in Fig. 1. There are two entities, the data owner (client) and the service provider (server). To verify the correctness of the outliers returned by the server, the client is equipped with an *auditor* component which can be viewed as a “black box” software. In particular, the auditor component consists of two modules, namely the *injector* module and the *tester* module. The injector module, which can be executed on a computer desktop, generates a set of artificial records (ARs) offline. The ARs are used to construct the *artificial outliers* (AOs) and *artificial non-outliers* (ANOs). The ARs are inserted into the original database and sent to the server. Meanwhile, the injector module produces a small piece of *auxiliary information* (including a hash function, three constants, and the number of AOs and ANOs). The auxiliary information is maintained at the *tester* side for verification purpose. After outsourcing

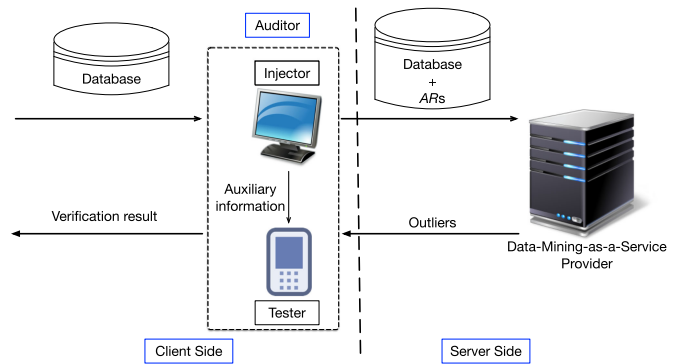


Fig. 1. The authentication architecture.

the dataset and the outlier detection task, the client receives and verifies the outliers returned by the server by executing the *tester* module. In particular, the tester module analyzes the returned outliers against the AOs and ANOs, and quantifies the probabilistic guarantee of the result correctness. We show that incorrect answers from the server can be caught with high confidence by utilizing a small number of AOs and ANOs, even for large datasets. This makes it feasible to run verification on the resource-constrained devices such as smart phones. We make the following contributions:

- 1) We design an efficient authentication approach that generates AOs and ANOs without any need to do outlier mining of the original dataset. One major challenge is that inserting any artificial record into the dataset may change the (non-)outlierness of the original records. We design a novel AOs/ANOs construction algorithm that does not eliminate any true outlier. We also discuss how to remove the false positive outliers (i.e., the original non-outliers that become outliers in the dataset with AOs/ANOs) introduced by AOs/ANOs and recover the true outliers efficiently. Formal analysis shows that a small number of AOs/ANOs can verify the result returned by the *Class I* server with high probabilistic guarantee.
- 2) To incentivize the *Class II* server to behave honestly, we design a game theoretic approach to decide the appropriate parameter values for the authentication setting, so that a rationale server should always return correct mining results to obtain the most payoff.
- 3) We discuss the *authentication-aware* cheating behaviors by the *Class III* server. As a countermeasure, we propose two approaches to catch the *Class III* server.
- 4) We run an extensive set of experiments on two datasets to demonstrate the efficiency and the effectiveness of our authentication approach. The experimental results show that the AO and ANO construction takes at most 1 second on the client side. Furthermore, the mining overhead at the server side is no more than 1.2 percent.

An earlier and short version of this paper was published in the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) [26]. The short version only considers the *Class I* server. In this paper, we extend the short version significantly with: (1) a game theoretic approach that analyzes the *Class II* server’s behaviors and incentivizes the server to return

correct mining results; and (2) the security analysis of our verification approach against the *Class III* server.

The paper is organized as following. Section 2 introduces the preliminaries. Section 3 presents our *AR*-based approach to construct *AOs* and *ANOs* to catch the *Class I* server. Section 4 discusses the strategic game to incentivize the *Class II* server to do mining and return correct results honestly. Section 5 analyzes the security against the cheating behaviors by the *Class III* server trying to escape from the *AR*-based authentication, assuming that it possesses the details of the authentication mechanism. Section 6 presents our experimental results. Section 7 discusses related work. Section 8 concludes the paper.

2 PRELIMINARIES

2.1 Distance-Based Outlier Mining

A variety of definitions of outliers, including distance-based outliers [21] and density-based outliers [8], have been proposed recently. In this paper, we focus on distance-based outliers, due to the properties that it works well for datasets of any dimension, and it does not assume that the data follows a standard distribution [21]. Specifically, an object t in the dataset D is a (p, d) -outlier if it is at least of distance d away from at least p proportion of all objects in D . In the rest of this paper, we use *outlier* and (p, d) -outlier interchangeably. If an object does not satisfy the condition, we call it a *non-outlier*. We measure the distance between two objects based the *euclidean distance*. In specific, given two records $t(a_1, \dots, a_k)$ and $t'(a'_1, \dots, a'_k)$, $dist(t, t') = \sqrt{\sum_{i=1}^k (a_i - a'_i)^2}$.

2.2 Outsourcing Setting

We consider the outsourcing setting that consists of two parties, the *data owner (client)* who possesses a dataset D and aims to find outliers in D , and the *service provider (server)* that provides distance-based outlier mining as the service. When the client outsources the dataset to the server, she communicates with the server regarding the parameter setting of p and d values for (p, d) -outlier mining. She can either choose to re-configure the p and d parameters by herself or follow the parameter settings by the server. The server executes outlier mining according to the parameter setting. We assume the server finds exact outliers instead of approximate ones [22], [37].

2.3 Adversary Model

In this paper, we adopt the taxonomy of [3] of the attackers to consider three types of servers that possess different knowledge of the authentication approach.

- The *Class I (ordinary outsider)* server that is not aware of the authentication method, but is curious to find out if there is any authentication mechanism.
- The *Class II (intelligent outsider)* server that is aware of the fact that the client may perform result correctness authentication. However, it does not have the details of the authentication approach.
- The *Class III (knowledgeable insider)* server that knows the details of the authentication approach and may try to escape the verification by returning well-designed incorrect answer intentionally.

2.4 Authentication Goal

Let O be the set of outliers in the outsourced dataset D , and O^S be the set of outliers returned by the server. We define the *precision* of O as $P = \frac{|O \cap O^S|}{|O^S|}$ and the *recall* as $R = \frac{|O \cap O^S|}{|O|}$. Intuitively, precision measures the fraction of returned outliers that are correct, while recall quantifies the fraction of correct outliers that are returned. The aim of our authentication approach is to catch any wrong answer, i.e., $P < 1$ or $R < 1$, with a high probability. Next, we formally define the authentication goal as (α, a) -completeness and (β, b) -soundness.

Definition 2.1. Given a dataset D and a verification method M , let pr_p and pr_r be the probabilities that M catches the server that returns the result of precision $P \leq b$ and recall $R \leq a$ respectively, where $a, b \in [0, 1]$ are given thresholds. We say M can verify (α, a) -completeness if $pr_r \geq \alpha$, and can verify (β, b) -soundness if $pr_p \geq \beta$, where $\alpha, \beta \in [0, 1]$ are also given threshold values.

If the client catches the server's cheating behavior, then the client is 100 percent sure that O^S is either unsound or incomplete. Otherwise, the client believes that O^S satisfies that: (1) $R > a$ with probability α ; and (2) $P > b$ with probability β .

2.5 Game Theory

A strategic game is a model of interacting decision-makers. The game involves a set of players. Each player has a set of possible actions. For each player, the preferences/payoffs over the set of action profiles are pre-defined. Each player chooses his/her action once and for all, and the players choose their actions "simultaneously" without any player being informed. A *Nash Equilibrium* is an action profile A^* with the property that no player can get more payoff by choosing an action different from her action in A^* , given that every other player adheres to A^* [33]. Intuitively, a Nash Equilibrium corresponds to a *steady state*. If, whenever the game is played, the action profile is the same as the Nash Equilibrium A^* , then no player has a reason to choose any action different from his/her component of A^* .

3 AR-BASED APPROACH TO CATCH CLASS I SERVER

The key idea of our authentication framework is that the client constructs a set of artificial records (*ARs*) before outsourcing. Let ΔD be the artificial records. Given the original dataset D , each record in ΔD is either an *artificial outlier (AO)* or an *artificial non-outlier (ANO)*. The client combines D with ΔD to get D^+ and sends D^+ to the server. If the server conducts the outlier mining faithfully, it should return all *AOs* but no *ANO*.

In this section, we focus on the *Class I* server that is not aware of the authentication procedure and thus cannot distinguish *ARs* from the original records. Therefore, the client is able to obtain a probabilistic guarantee of the soundness and completeness of the returned outliers by comparing them with the *AOs* and *ANOs*. Next, we first discuss how to construct *ANOs* and *AOs* in Sections 3.1 and 3.2. After that, we prove that the *AR*-based approach preserves the (non) outliers in the original dataset in Section 3.3. Sections 3.4 and 3.5 discuss the auxiliary data and the authentication procedures at the client side.

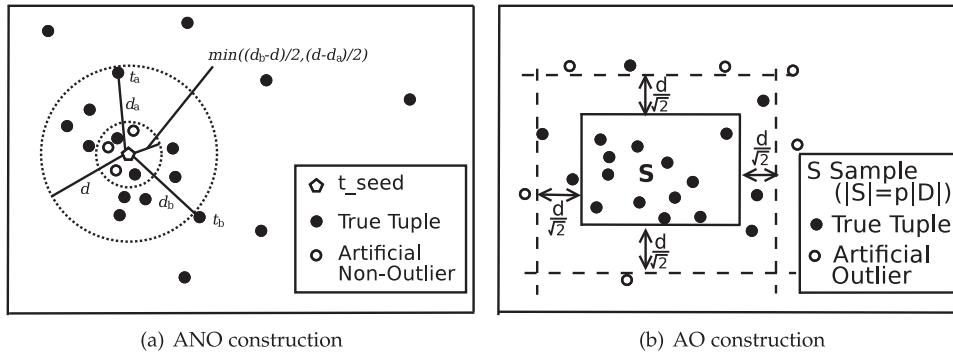


Fig. 2. Construction of ANOs and AOs.

3.1 Construction of Artificial Non-Outliers (ANOs)

Before we introduce the ANO construction procedures, we define *close records* and *distant records* first.

Definition 3.1. Given a dataset D and a record t , the close records of t with regard to d are $T_L(t, d) = \{t' | t' \in D, \text{dist}(t, t') < d\}$, while the distant records are $T_U(t, d) = \{t' | t' \in D, \text{dist}(t, t') > d\}$.

Intuitively, $T_L(t, d)$ is the set of records in D whose distance to t is smaller than d , while $T_U(t, d)$ stores the set of records that are at least d distance away from t . Next, we define the *farthest close neighbor* and *closest distant neighbor* respective.

Definition 3.2. Given a dataset D , a record t and a distance threshold d , a record $t' \in T_L(t, d)$ is the farthest close neighbor of record t , if the distance between t and t' is the largest among all records in $T_L(t, d)$. Similarly, a record $t' \in T_U(t, d)$ is the closest distant neighbor of record t , if the distance between t and t' is the smallest among all records in $T_U(t, d)$.

Definition 3.3. Given the dataset D of r dimensions, a record $t \in D$ and the distance threshold d , let $t_a \in D$ be the farthest close neighbor of t , and $t_b \in D$ be the closest distant neighbor of t . Let $d_a = \text{dist}(t, t_a)$, and $d_b = \text{dist}(t, t_b)$. Let Q be an r -sphere with t as the centroid, and $\min(\frac{d-d_a}{2}, \frac{d_b-d}{2})$ as the radius, where d is the distance parameter of (p, d) -outlier mining. Then we say any record $t' \in D$ is a close record to t if it falls in Q .

Next, we show that the close records of t have the same distance property as t .

Lemma 3.1. Given a record t and a close record t_c of t , for any record $t' \neq t$, it must be true that: (1) if $\text{dist}(t, t') < d$, then $\text{dist}(t_c, t') < d$; and (2) if $\text{dist}(t, t') > d$, then $\text{dist}(t_c, t') > d$.

Proof. Since t_a is the farthest close neighbor of t , for any t' s.t. $\text{dist}(t, t') < d$, we have $\text{dist}(t, t') \leq d_a < d$, and $\text{dist}(t, t_c) < \frac{d-d_a}{2}$, leading to $\text{dist}(t', t_c) \leq \text{dist}(t, t') + \text{dist}(t, t_c) < d_a + \frac{d-d_a}{2} = \frac{d+d_a}{2}$. Since $d_a < d$, then it must be true that $\text{dist}(t_c, t') < d$. Similarly, we have $\text{dist}(t, t') \geq d_b > d$, and $\text{dist}(t, t_c) < \frac{d_b-d}{2}$. Thus, $\text{dist}(t_c, t') \geq \text{dist}(t, t') - \text{dist}(t, t_c) > d_b - \frac{d_b-d}{2} = \frac{d_b+d}{2}$. Since $d_b > d$, then it must be true that $\text{dist}(t_c, t') > d$. \square

Note that Lemma 3.1 holds no matter t' is an outlier or non-outlier in the dataset D . Based on Lemma 3.1, we use Theorem 3.1 to prove that any record close to record t always has the same non-outlierness as t .

Theorem 3.1. Given a dataset D and any record $t \in D$ that is a non- (p, d) -outlier, any close record of t must be a non- (p, d) -outlier in D .

Proof. Let t_c be a close record of a true record t in D . It is straightforward from Lemma 3.1 that t_c and t have the same number of records whose distance is greater than d in D^+ . If t is a (p', d) -outlier in D^+ , t_c must be a (p', d) -outlier in D^+ . On the other hand, if t is a (p', d) -non-outlier in D^+ , t_c must be a (p', d) -non-outlier in D^+ . \square

Theorem 3.1 provides us the guidance to construct ANOs. In particular, we first pick a *seed* record t_{seed} that is a non-outlier record in the original dataset D . Then we construct a set of artificial records as the close records of t_{seed} , and take these artificial records as ANOs. Fig. 2a illustrates the construction procedure of ANO in a 2-dimensional dataset.

To pick t_{seed} , we repeatedly pick a record from D randomly, and check its non-outlierness, until a non-outlier record is selected. The probability that t_{seed} will be picked at the x th trial is

$$g(x) = \left(1 - \frac{f_{to}}{n}\right) \left(\frac{f_{to}}{n}\right)^{x-1},$$

where f_{to} is the number of outliers, and n is the number of records in D respectively. It is straightforward that $1 \leq x \leq n - f_{to}$. We define $\phi = \frac{f_{to}}{n}$. Then $g(x) = (1 - \phi)\phi^{x-1}$, where $1 \leq x \leq n - n\phi$. The expected value of x equals to

$$E(x) = \frac{(n - \phi n)\phi^{n-\phi n+1} - (n - \phi n + 1)\phi^{n-\phi n} + 1}{(\phi - 1)^2}.$$

As usually the outliers are only a small portion of the whole dataset, ϕ is a small number (e.g., $\phi = 0.05\%$ [35]). Therefore, $E(x) \approx 1$. In other words, it is highly likely that t_{seed} can be picked by the first random trial.

After t_{seed} is identified, we can construct ANOs by constructing the r -sphere Q (defined in Definition 3.3) of t_{seed} , and picking any record in Q as a ANO. The computation of t_{seed} needs to traverse D once. It requires a scan of the dataset D to identify t_{seed} , and takes another round to determine the radius of the r -sphere Q (defined in Definition 3.3). Therefore, we can construct ANOs by two passes of D .

3.2 Construction of Artificial Outliers (AO)

A seemingly straightforward way to construct AOs is similar to the construction of ANOs: we first find a seed outlier from

the dataset D and then construct artificial records that are close to the seed as AOs. However, this approach may be prohibitively expensive, since the outliers are rare and finding an outlier may require mining of the original dataset. Our goal is to construct AOs without mining the original dataset to find any outlier. In the following, we discuss the details of how to construct AOs efficiently.

Our construction procedure relies on the definition of *distant* records.

Definition 3.4. Given a r -dimension dataset D and a set of records $S \subseteq D$, we say a record $t \notin S$ is a distant record of S if for each record $t' \in S$, $\text{dist}(t, t') > d$, where d is the parameter setting of (p, d) -outlier mining.

We have the following lemma to show a important property of distant records.

Lemma 3.2. Given a r -dimension dataset D and a set of records $S \subseteq D$, let \min_i and \max_i be the minimum and maximum value of the i th ($1 \leq i \leq r$) attribute of the records in S . Then any record $t \notin S$ is a distant record of S if there are k ($1 \leq k \leq r$) attributes such that on each attribute A_i ($1 \leq i \leq k$), $t[A_i] < (\min_i - \frac{d}{\sqrt{k}})$ or $t[A_i] > (\max_i + \frac{d}{\sqrt{k}})$, where $t[A_i]$ is the i th attribute value of t .

The correctness of Lemma 3.3 is straightforward. Next, we use Theorem 3.2 to show that (p, d) -outliers can be generated from the distant records.

Theorem 3.2. Given the dataset D and a set of records $S \subseteq D$, any distant record t of S must be a (p, d) -outlier if $|S| \geq p|D|$.

The correctness of Theorem 3.2 is straightforward. For any distant record t , there must exist a set of records $S \subseteq D$, whose size is greater than p fraction of the size of D , such that the distance between t and any record $t' \in S$ must be greater than d . Therefore, t must be a (p, d) -outlier. Based on Theorem 3.2, we design the AO construction procedure as follows.

First, the client picks a sample S of size $\lceil p|D| \rceil$ records randomly from D . Second, the client treats S as an r -dimension hypercube \mathcal{R} . The range $[\min_i, \max_i]$ represents the edge at the i th dimension of \mathcal{R} . Then the client randomly picks $k \leq r$ dimensions (possibly $k = 1$) of \mathcal{R} . Last, the client expands the picked k dimensions of \mathcal{R} by $\frac{d}{\sqrt{k}}$ (i.e., change the minimum and maximum value of the i th attribute to be $\min_i - \frac{d}{\sqrt{k}}$ and $\max_i + \frac{d}{\sqrt{k}}$). Let the expanded hypercube be \mathcal{R}' . Then any record t_{ao} that is created outside of \mathcal{R}' must be a (p, d) -outlier of D . Fig. 2b illustrates the construction procedure in a 2-dimensional dataset. How to decide f_{ao} will be discussed in Section 3.5.1. The complexity of AO construction is $O(n)$, where n is the size of D .

3.3 Preservation of (Non)Outlierness

One issue of inserting ARs into D is that (p, d) -outliers in the original dataset D may not be (p, d) -outliers in $D^+ = D \cup \Delta D$ anymore, as inserting records into D may change the (non) outlierness of some original records in D . This may ruin the authentication method as even the server executes the outlier mining and returns the result faithfully, it will be wrongly caught by the AR-based verification approach. Therefore, the client must make sure that all (p, d) -outliers

in D are also outliers in D^+ . In this section, we discuss the followings:

- How to ensure that the true (p, d) -outliers in D are still recognized as outliers in D^+ ; and
- How to eliminate the *false positive* outliers (i.e., the records that are not (p, d) -outliers in D but become outliers in D^+).

Preservation of True AOs. First, we show how to make sure that the (p, d) -outliers in D are still outliers in D^+ by only using a different p value in (p, d) -outlier mining.

Theorem 3.3. Given a dataset D and a set of AOs and ANOs constructed from the set of artificial records ARs, any (p, d) -outlier in D must be a (p_1, d) -outlier in $D^+ = D \cup \Delta D$, where

$$p_1 = \frac{p|D|}{|D^+|}. \quad (1)$$

Proof. For a true record $t \in D$, let x and y be the number of original and artificial records (including both AOs and ANOs) whose distance to t is greater than d in D^+ . Now we prove that it must be true that $\frac{x+y}{|D^+|} \geq p_1 = \frac{p|D|}{|D^+|}$. This can be proven by the following. Since record t is a (p, d) -outlier in D , it must be true that $x \geq p|D|$. This naturally leads to that $\frac{x+y}{|D^+|} \geq \frac{p|D|}{|D^+|}$, with $x \geq 0$. \square

Following Theorem 3.3, instead of requesting for (p, d) -outliers in the outsourced dataset D^+ , the client asks for (p_1, d) -outliers in D^+ , where p_1 is defined in Formula (1). All (p, d) -outliers in D must appear in the answer if the server is honest. Note that all AOs must appear in (p_1, d) -outliers of D^+ too.

Elimination of False Positive Outliers. It is not necessary that all (p_1, d) -outliers in D^+ must be (p, d) -outliers in D . In the words, by asking for (p_1, d) -outliers on D^+ , the server may return some *false positive* outliers. To eliminate those false positives, we have:

Theorem 3.4. Given a dataset D , let ΔD be the set of ARs that are used to construct AOs and ANOs. Then any (p_2, d) -outlier in $D^+ = D \cup \Delta D$ must be a (p, d) -outlier in D , where

$$p_2 = \frac{p|D| + |\Delta D|}{|D^+|}. \quad (2)$$

Proof. For a record $t \in D^+$, let x and y be the number of original and artificial records (including both AOs and ANOs) whose distance to t is greater than d in D^+ . Since the x true records must exist in D , we aim to prove that $\frac{x}{|D|} \geq p$. This can be proven as follows. Since t is a (p_2, d) -outlier in D^+ , it must be true that $\frac{x+y}{|D^+|} \geq p_2$. In other words, $x + y \geq p|D| + |\Delta D|$. Since $y \leq |\Delta D|$, it must be true that $x \geq p|D|$. Therefore, the theorem holds. \square

Following Theorem 3.4, all the constructed ANOs must be (p_2, d) -non-outliers in D^+ .

Fig. 3 illustrates the relationship among (p_1, d) - and (p_2, d) - outliers (p_1 and p_2 are defined by Equations (1) and (2) respectively) in D^+ and (p, d) -outliers in D . For a given record $t \in D$, let ρ be the fraction of records in $D^+ = D \cup \Delta D$ whose distance to t is greater than d , where d is the parameter for (p, d) -outlier mining. Then t must fall into one of the following three categories:

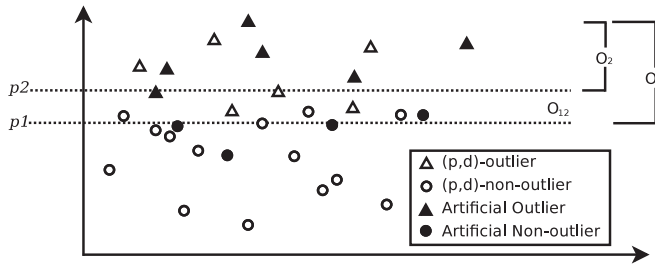


Fig. 3. (p_1, d) -outliers and (p_2, d) -outliers in D^+ versus (p, d) -outlier in D ; O_1 : (p_1, d) -outliers in D^+ , O_2 : (p_2, d) -outliers in D^+ , O_{12} : $O_1 - O_2$.

- t is a (p, d) -outlier in D , if $\rho \geq p_2 = \frac{p|D| + \Delta D}{|D^+|}$;
- t is a (p, d) -non-outlier in D , if $\rho < p_1 = \frac{p|D|}{|D^+|}$;
- t is either a (p, d) -outlier or a (p, d) -non-outlier in D , otherwise.

Based on both Theorems 3.3 and 3.4, the outsourcing and the authentication procedures are designed as the following. The client constructs $D^+ = D \cup \Delta D$ and outsources D^+ to the server. She sends two mining requests to the server for (p_1, d) -outliers and (p_2, d) -outliers, where $p_1 = \frac{p|D|}{|D^+|}$ and $p_2 = \frac{p|D| + \Delta D}{|D^+|}$. Let O_1 and O_2 be the set of outliers received from the two requests separately. It is worth noting that $O_2 \subseteq O_1$. Therefore, the client can get both (p_1, d) -outliers and (p_2, d) -outliers by outsourcing the task only once.

3.4 Auxiliary Data for Authentication

Auxiliary Data Sent to the Server. Before the client sends out her dataset, she generates two digests, namely dig_a and dig_b , for each record $t \in D^+$ with a cryptographic hash function. Let s be the secret key of the client, and H denote the signing function of a message authentication code (MAC), e.g., the Hash-based message authentication code (HMAC). Given a record $t(a_1, \dots, a_r)$, we have $dig_a = H(s, t)$, and

$$eqalignndig_b = \begin{cases} H(c_1, dig_a), & \text{If } t \text{ is a true record in } D; \\ H(c_2, dig_a), & \text{If } t \text{ is an AO}; \\ H(c_3, dig_a), & \text{If } t \text{ is an ANO}, \end{cases}$$

where c_1, c_2 and c_3 are three unique constant values kept secret at the client side.

Intuitively, dig_a helps the client to check if the server has modified any record in D^+ , while dig_b can be used to distinguish the original records from the AOs and ANOs.

After completing the computation of digests, the client sends D^+ to the server. Each record is associated with its two digests dig_a and dig_b . When the server returns any outlier to the client, it is required to return the two digests too.

Auxiliary Data Maintained at the Client Side. The client keeps the private key s locally, and maintains the number of AOs and ANOs, and the three constants c_1, c_2 , and c_3 that are used to construct the digests. It is straightforward that the space overhead of these auxiliary information is negligible. For each outlier record t returned by the server, the client computes its digest dig_a , and the three signatures $dig_b^1 = H(dig_a * c_1)$, $dig_b^2 = H(dig_a * c_2)$, and $dig_b^3 = H(dig_a * c_3)$, by using the three constants c_1, c_2 , and c_3 that it has stored locally. Then by comparing the digests $dig_b^1, dig_b^2, dig_b^3$ against the dig_b that is attached to t , the client can distinguish whether t is a original record in D , an AO or an ANO.

3.5 Authentication and Post-Processing at Client Side

After receiving O_1 and O_2 from the server, the client checks the integrity and post-processes the result as the following.

3.5.1 Authentication at Client Side

The client runs the 2-phase verification procedure to check the soundness and completeness of the result.

Phase-1. For each record $t \in O_1$ or $t \in O_2$, the client recomputes the digest dig_a by applying the stored hash function H (Section 3.4) on t . If the computed digest does match the one that is associated with t , the client concludes that the server has returned records that do not exist in the outsourced database, and thus fails to pass the verification.

Phase-2. If the server passes the phase-1 verification, the client further computes three digests: $dig_b^1 = H(dig_a * c_1)$, $dig_b^2 = H(dig_a * c_2)$, and $dig_b^3 = H(dig_a * c_3)$, by using the three constants c_1, c_2 and c_3 that the client stores locally. By comparing these digests with the digest dig_b associated with t , the client can identify whether the returned tuple is a true record, an AO, or an ANO. Then the client verifies the completeness and soundness respectively.

Completeness Authentication. To verify whether the server has returned all true outliers, the client checks whether every AO is included in O_1 . If not, the client catches the incomplete outlier answer with 100 percent; otherwise, the client trusts that the recall R of the result is at least α with probability $pr_r = 1 - \alpha^{f_{ao}}$, where f_{ao} is the number of artificial outliers. This is because for each AO in D^+ , the probability that the server includes it in O_1 is α . The probability that the server avoids being caught by the client, i.e., the probability that the server returns all AOs in O_1 , is $\alpha^{f_{ao}}$. Therefore, to satisfy (α, a) -completeness (i.e., $pr_r \geq a$), it is required that

$$f_{ao} = \lceil \log_{\alpha}(1 - a) \rceil. \quad (3)$$

Soundness Authentication. For the soundness authentication, the client checks whether if any ANO appears in O_2 . If it does, the client catches the incorrect answer with 100 percent; otherwise, the client believes that the precision P of the result returned by the server is at least β with probability $pr_p = 1 - \beta^{f_{ano}}$, where f_{ano} is the number of artificial non-outliers. Similar to the reasoning in the completeness authentication, the probability that the server does not return any ANO in O_2 is $\beta^{f_{ano}}$. Thus to meet the (β, b) -soundness (i.e., $pr_p \geq b$) requirement, f_{ano} must satisfy that

$$f_{ano} = \lceil \log_{\beta}(1 - b) \rceil. \quad (4)$$

Equations (3) and (4) show that f_{ao} and f_{ano} (the number of AOs and ANOs) are independent of the size of D . Therefore, our authentication framework would be especially efficient for large datasets. Furthermore, it does not need large number of AOs and ANOs to provide high soundness and completeness guarantee. For instance, when $\alpha = 0.99$ (i.e., the server misses 1 percent of the outliers) and $a = 0.99$ (i.e., the probability to catch such answer is at least 0.99), it only needs 459 AOs to verify (0.99, 0.99)-completeness.

Overhead Analysis. The complexity of soundness and completeness authentication is $O(f_{ao} + f_{ano})$. Our empirical study shows that f_{ao} and f_{ano} are relatively small even for

TABLE 1
Notations

Notation	Meaning
c_v	Client's cost for <i>AR</i> -based authentication
c_m	Server's cost for faithful outlier mining
p_c	Server's penalty if the cheating is caught
p_m	Server's payment for mining
u_m	Client's benefit gained from correct outlier mining result
$\epsilon \in [0, 1]$	Probability of catching server by <i>AR</i> -based authentication
$\eta \in [0, 1]$	Result correctness belief for trust-without-verification
$\lambda \in [0, 1]$	Result correctness belief for pass of verification
$\sigma \in [0, 1]$	Fraction of correct outliers that are returned by the server

large datasets (Section 6). This enables the client to accomplish authentication on resource-constrained devices (e.g., smart phones).

3.5.2 Recovery of True (p, d) -Outliers at Client Side

Since the returned (p_1, d) -outliers O_1 and (p_2, d) -outliers O_2 may contain some false positive records that are not (p, d) -outliers in D , the client needs to recover the original (p, d) -outliers in D from O_1 and O_2 . To accomplish this, first, the client excludes all *AOs* (if there is any) from O_2 (how to distinguish original records from *AOs*, and *ANOs* is discussed in Section 3.5.1). Let the remaining (p_2, d) -outliers be O'_2 . Second, the client checks each record in $O_1 \setminus O_2$ against D , and keeps those that are (p, d) -outliers in D . Let these records be O_{12} . Then $O_{12} \cup O'_2$ are the set of original (p, d) -outliers in D . As will shown in the Experiment section (Section 6), the records in $O_1 \setminus O_2$ take a negligible portion of D (less than 0.2 percent). Therefore, the complexity of outlier recovery at the client side is $O(|D|)$.

4 A GAME THEORETIC APPROACH TO INCENTIVIZE CLASS II SERVER

While our *AR*-based authentication approach enables the client to catch the *Class I* server with high probability, a *Class II* server that is aware of the probabilistic authentication strategy may still make the most outcome by performing cheaper outlier mining computations and returning incorrect results. In this section, we aim at incentivizing the *Class II* server not to cheat on the outlier mining computations. A possible solution is to build a strategic game [48] between the client and server, in which a Nash equilibrium [36] exists only when the server's action is to return correct mining results. Based on the theory of rational choice, due to the Nash Equilibrium, a rational server will always play the game honestly, if he can not gain more payoff by switching to cheating. We emphasize that even though we focus on the outsourced outlier mining, our incentive game model can be applied to other types of outsourced computations.

There are two players, i.e., the server and the client, in this game. Each player has the following actions.

- *Client*
 - *Verify*: The client uses the *AR*-based approach to authenticate the returned outlier mining results.
 - *Trust-without-verification*: The client simply accepts the results returned by the server. With a certain level of belief, the client trusts the result to be

TABLE 2
Payoff Matrix for the Client and Server

		Not Cheat	Cheat
Verify	Catch	N/A	$(\sigma u_m + p_c - c_v, -\sigma c_m - p_c)$
	Not Catch	$(\lambda u_m - c_v - p_m, p_m - c_m)$	$(\sigma u_m - c_v - p_m, p_m - \sigma c_m)$
Not Verify		$(\eta u_m - p_m, p_m - c_m)$	$(\sigma u_m - p_m, p_m - \sigma c_m)$

correct. The belief can be obtained based on the server's reputation.

If the client accepts the server's mining results (either after verification or without verification), the client pays the server for its mining efforts. The payment is considered as the server's payoff.

- *Server*
 - *Cheat*: The server returns wrong outliers to the client. If the cheating behavior is caught by the client, the server receives no payment from the client. Instead, it pays the penalty for the cheating.
 - *Honest play*: The server executes mining faithfully and returns the correct outliers to the client.

For the following discussions, we use the notations defined in Table 1.

Next, we present the payoff matrix (Table 2) for the client and server with various actions. We assume c_v , c_m , and p_c are in the monetary format. We also assume that the client's benefit gained from correct outlier mining result (i.e., u_m) can be measured in the same format as c_v , c_m , and p_c . Regarding the results that were considered as correct with probability η (λ , resp.), the client's benefit gained from the result is measured as ηu_m (λu_m , resp.). We use $(,)$ to denote the payoffs, where the left (right, resp.) value is the client's (server's resp.) total payoff by the specific actions. We explain the details of various payoffs as below.

- (*Verify, Not Cheat*): When the server does not cheat, there is no need that the client catches the server (with N/A in the corresponding *Catch* row). Thus the client trusts the returned outlier results with λ confidence and pays the full price for the server's mining efforts. The client's payoff is $\lambda u_m - c_v - p_m$, and the server's payoff is $p_m - c_m$.
- (*Verify, Cheat*): Due to the probabilistic property of the *AR*-based authentication approach, the client may not catch the server's cheating behaviors. Assume that the server returns σ ($0 < \sigma < 1$) fraction of the outliers, the client can still obtain σu_m utility, while the server's mining cost is σc_m . If the client catches the server (i.e., for the *Catch* case), the server needs to pay the penalty to the client. So the client's payoff is $\sigma u_m + p_c - c_v$, and the server's payoff is $-\sigma c_m - p_c$. If the client does not detect the cheating behavior (i.e., for the *Not Catch* case), the client still needs to pay the server for the mining. So the client's payoff is $\sigma u_m - c_v - p_m$, and the server's payoff is $p_m - \sigma c_m$.
- (*Not Verify, Not Cheat*): The client trusts the result correctness with probability η . Thus the client's payoff is $\eta u_m - p_m$, and the server's is $p_m - c_m$.

- (*Not Verify, Cheat*): As the client does not verify the result, the server receives the payment from the client, but the client does not benefit from the mining result. Thus, the client's payoff is $\sigma u_m - p_m$, and the server's is $p_m - \sigma c_m$.

Ideally, we want the action profile ($*$, Not Cheat) to be the steady state (Nash Equilibrium), where $*$ stands for any action for the client. This is because any Nash Equilibrium ($*$, Not Cheat) incentivizes the server to discover the outliers honestly. However, it is impossible to reach an equilibrium for the (Not Verify, Not Cheat) case, as the server can easily get more payoff by switching to Cheat. Thus, we can only get the desired equilibrium from (Verify, Not Cheat). According to the definition of Nash Equilibrium, an action profile is an equilibrium only if any player cannot gain more payoff by changing the action. Thus we require that

$$P_{(Verify, Not Cheat)}^{Server} \geq P_{(Verify, Cheat)}^{Server}, \quad (5)$$

and

$$P_{(Verify, Not Cheat)}^{Client} \geq P_{(Not Verify, Not Cheat)}^{Client}. \quad (6)$$

Based on the payoffs in Table 2, by doing the integral over σ , we have

$$\begin{aligned} P_{(Verify, Cheat)}^{Server} &= \int_0^\lambda \epsilon(-\sigma c_m - p_c) d\sigma \\ &\quad + \int_0^\lambda (1 - \epsilon)(p_m - \sigma c_m) d\sigma \\ &\quad + \int_\lambda^1 (p_m - \sigma c_m) d\sigma \\ &= -\epsilon \lambda p_c - \epsilon \lambda p_m + p_m - \frac{1}{2} c_m, \end{aligned} \quad (7)$$

and

$$P_{(Not Verify, Not Cheat)}^{Client} = \eta u_m - p_m. \quad (8)$$

The first part in Equation (7) measures the server's expected payoff when $\sigma < \lambda$ and its cheating is not detected by the client (the catching probability is ϵ). The second part evaluates the server's payoff when getting detected in cheating (probability is $1 - \epsilon$). Whereas the last part considers that the returned result contains more than λ fraction of correct result and is not detected by the AR-based approach.

While

$$P_{(Verify, Not Cheat)}^{Server} = p_m - c_m, \quad (9)$$

and

$$P_{(Verify, Not Cheat)}^{Client} = \lambda p_m - c_v - c_m. \quad (10)$$

According to the requirement, the action profile (*Verify, Not Cheat*) is a Nash Equilibrium if

$$\epsilon \lambda \geq \frac{c_m}{2(p_c + p_m)}, \quad (11)$$

and

$$\epsilon \geq \eta + \frac{c_v}{u_m}. \quad (12)$$

By combining Equations (11) and (12), we can get the required value of λ as

$$\lambda \geq \frac{c_m u_m}{2(p_m + p_c)(\eta u_m + c_v)}. \quad (13)$$

According to our AR-based verification approach, ϵ corresponds to α and β in the (α, a) -completeness and (β, b) -correctness model (Definition 2.1), while λ confirms to a and b in the same model. Therefore, in order to incentivize the *Class II* server to behave honestly, when constructing the AOs and ANOs, the client should make sure that

$$\alpha, \beta \geq \eta + \frac{c_v}{u_m}, \quad (14)$$

and

$$a, b \geq \frac{c_m u_m}{2(p_m + p_c)(\eta u_m + c_v)}. \quad (15)$$

5 SECURITY ANALYSIS AGAINST CLASS III SERVER

In this section, we focus on the *Class III* server that possesses the details of our AR-based authentication approach. We first discuss the security weakness of the AR-based approach. Then we present two robust verification approaches to catch the *Class III* server.

5.1 Weakness of AR-Based Authentication

We consider the *authentication-aware cheating* behaviors by the server: When the server is aware of the details of the authentication mechanism, it can identify (at least the candidates of) AOs and ANOs. Even though the server cheats in the outlier mining, it can avoid getting detected by returning all AOs and excluding all ANOs from the returned result.

Unfortunately, our AR-based approach cannot catch the authentication-aware cheating. When the server knows the details about how ARs are constructed, it is able to identify them from D^+ . In particular, it can find all AOs by two passes of D^+ . Because the p value of the (p, d) -outlierness is always very close to 1 in practice, the sample S used to construct AOs (Section 3) includes almost all records in D . Therefore, the constructed AOs will be the skyline points of D . Based on this, the server takes the records that have the maximum/minimum values of any attribute as AOs. On the other hand, due to the fact that all ANOs must be the *close* records (Definition 3.3) to a non-outlier record, the server can identify them by searching for non-outlier records that are *close* to at least one non-outlier. This requires the server to find all non-outliers, whose complexity is at least as high as the cost of mining all outliers. Though expensive, after finding the AOs and ANOs, the server is able to escape from the soundness and completeness authentication.

5.2 Catching Class III Server

Sampling-Based Probabilistic Approach. To catch the authentication-aware cheating, the client can randomly pick records from the original dataset as a *sample*, and verify the outlierness of the sample. To check the soundness and completeness of the result returned by the server, the client checks the outlierness of the samples against the result. Intuitively, a large sample is necessary in order to obtain a high soundness/completeness guarantee. This may not be affordable by the client with limited computational power. We conjecture that the complexity of catching the authentication-aware

cheating is as expensive as outlier mining itself, especially if the client requires a high result correctness probability.

Replication-Based Approach. The client can assign the outlier mining task to two servers, and verify the correctness by crosschecking the results from the two servers. To forbid collusion where the two servers return the same incorrect results, the client can create distrust between the servers by incentivizing them to betray their partner in the collusion coalition [16]. In specific, the client first takes deposit from both servers as the security for the delivery of correct answer. If any server is detected to return incorrect results, it loses the security deposit. Moreover, if one server is honest while the other is cheating, the cheating party's deposit will be taken by the honest one. Theoretical game theory analysis [16] shows that by setting appropriate deposit amounts, both rational servers will choose not to cheat to receive the maximum payoff. In this way, it is guaranteed that the *Class III* server returns correct outlier mining results.

6 EXPERIMENTAL EVALUATION

We conducted an extensive set of experiments to evaluate both the robustness and performance of our *AR*-based authentication approach. In particular, we measured: (1) the soundness and completeness guarantee; (2) the verification overhead at the client side, which includes a) the construction time of *AOs* and *ANOs*, b) the verification time to based on *AOs* and *ANOs*, and c) the time of examining the outlier-ness of records to eliminate false positives; (3) the mining overhead at the server side.

Setup. In our experiment, we use two datasets, the *Letter* dataset¹ from UCI MLC++ Library that contains 20k records, and the *KDDCUP* dataset² that contains 100k records. The *Letter* dataset has 16 numerical (integer) attributes. The *KDDCUP* dataset contains 41 (numerical or categorical) attributes. In our experiments, we use five numerical attributes, including *duration*, *dst_bytes*, *number_access_files*, *count*, and *error_rate* attributes, of the *KDDCUP* dataset. For *Letter* dataset, we used $p = 0.8$ and $d = 15$ that return 1 percent of the records as outliers, while for *KDDCUP* dataset, we used $p = 0.99$ and $d = 5000$ that return 2 percent of the records as outliers. All of our experiments are evaluated on a PC with a 2.4 GHz Intel Core 2 Duo CPU and 4GB RAM running Windows 7. We implemented the algorithm in Java.

ADS-Based Deterministic Authentication Approach. We implement the deterministic authentication approach that can verify the result correctness with 100 percent certainty. We design the deterministic approach by using the *Merkle B-tree* [24] as the *authenticated data structure*. By using the Merkle B-tree, the server constructs a verification object and sends the VO to the client. From the VO, the client checks if the server has tampered with the outsourced dataset. After that, the client verifies the soundness and completeness by computing the distance between every pair of tuples.

6.1 Robustness of *AR*-Based Approach

We simulate the incomplete result by removing 5, 10, 15, 20, and 25 percent outliers randomly (i.e., $a = 95, 90, 85, 80,$ and

75 percent), and generate the incorrect result as inserting 5, 10, 15, 20, and 25 percent non-outliers into the result (i.e., $b = 95, 90, 85, 80,$ and 75 percent). Then, we measure the probability of catching these incomplete and unsound results of our approach as the following: we repeat 500 trials on the *Letter* dataset and 100 trials on the *KDDCUP* dataset. In each trial, we construct a certain number of *AOs* and *ANOs* and compare the result against them.

For completeness verification, we check if all the *AOs* are included in the server's result. If at least one *AO* is missing, we take the trial as a *successful detection*. Similarly, for soundness verification, we check if the result contains any *ANO*, and take the trial as a detection if it does. After we finish all trials, we calculate the detection probability as the $pr_d = \frac{n_{det}}{n_{tr}}$, where n_{det} is the number of successful trials and n_{tr} is the total number of trials.

First, we measured the detection probability of incomplete answer. Fig. 4a shows the result on the *Letter* dataset. We observe that the detection probability is always higher than the required α value (i.e., the probability threshold). We have the same observation on the *KDDCUP* dataset (Fig. 4b). We also measured the detection probability of soundness violations and show the results in Figs. 4c & 4d for *Letter* dataset and *KDDCUP* dataset respectively. It can be seen that the detection probability of unsound result is always higher than the required β value, i.e., the soundness catching probability threshold. This proves the robustness of our *AR*-based approach for both completeness and soundness verification.

6.2 Cost Analysis at Client Side

First, we measured the time performance of the *AR*-based approach on the *Letter* dataset. Fig. 5a shows the *AO* construction time. We observe that the *AO* construction is extremely fast, as it only takes 0.012 seconds even when $\alpha = 0.95$ and $a = 0.95$. Furthermore, when α and a values increase, *AO* construction time increases too, but only slightly. Fig. 5b shows the *ANO* construction time on *Letter* dataset. It takes more time than *AO* construction since it needs to find the t_{seed} . Nevertheless, it is still very efficient; it only needs 0.022 second at most even when $\beta = 0.95$ and $b = 0.95$. Compared with the ADS-based deterministic authentication approach (around 0.18 second to construct the authenticated data structure for both completeness and soundness verification on the *Letter* dataset), the construction time of *AOs* and *ANOs* is negligible. We also measured the verification time at the client side. Fig. 5c shows the time of completeness verification on *Letter* dataset. We observed that the time grows when α and a increase. This is straightforward as higher α and a require larger number of *AOs*. In comparison with the ADS-based deterministic approach, the *AR*-based authentication approach dramatically saves the verification cost at the client side. For example, the ADS-based deterministic approach demands 131 seconds to check the result completeness. Whereas, the *AR*-based approach only needs at most 0.008 second. Fig. 5d shows the time of soundness verification. Contrary to the completeness verification, the soundness verification time decreases with the growth of β and b . This is because with a larger b value, there are fewer original non-outliers inserted as unsound answer (for simulation). Even though a larger b value demands more *ANOs*, the number of inserted original non-outliers decreases faster than that of *ANOs*. This leads

1. <http://www.sgi.com/tech/mlc/db/letter.all>

2. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

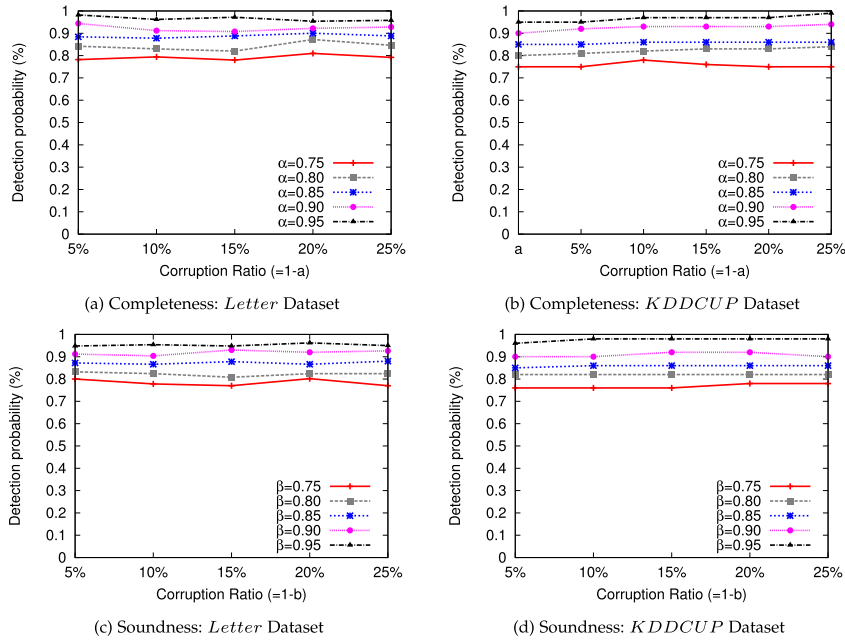


Fig. 4. Robustness of *AR*-based approach.

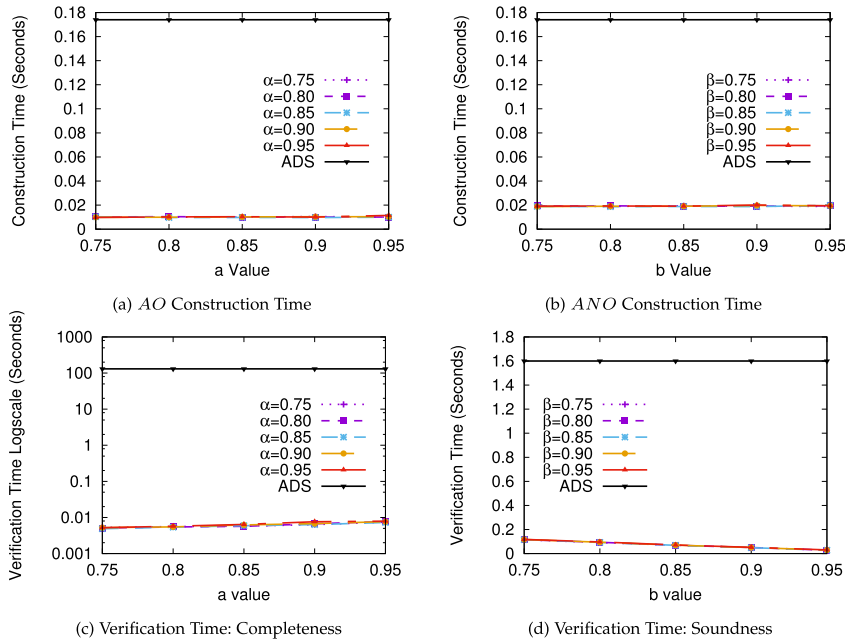


Fig. 5. *AO*&*ANO* construction time and verification time, *Letter* dataset.

to the decreasing number of outliers (including real non-outliers and *ANOs*) that the client receives, and consequently less verification time. Compared with the *ADS*-based deterministic approach, the *ANO*-based soundness verification is at least 10 times more efficient.

Second, we measured the time performance on *KDDCUP* dataset and compared it with the baseline approach. We note that the *ADS*-based deterministic approach cannot finish the completeness verification within 100 hours. Thus we note “N/A” in the corresponding cell of Table 3. We found that both the *AO* & *ANO* construction and verification time are more than that of the *Letter* dataset, as the size of the *KDDCUP* dataset is four times larger than that of the *Letter* dataset. However, the *AO/ANO* construction is still fast; *AO* construction only takes 0.165 second at most, while *ANO*

construction only takes 0.035 seconds at most. The *ADS*-based deterministic approach takes 15.341 seconds to build the authenticated data structure, which is 100 times slower than the *AR*-based approach. The *AR*-based approach is also significantly more efficient than the *ADS*-based deterministic approach. In particular, the completeness verification takes at most 0.042 second, while the correctness verification finishes within 0.5 second. However, the *ADS*-based deterministic approach needs more than 10 hours to check the correctness, and cannot finish the completeness verification within 100 hours. This proves that our *AR*-based approach is in particular suitable for the verification under resource-constrained environment (e.g., on smart phones). With little sacrifice on the correctness guarantee, the client saves significantly for verification.

TABLE 3
AO&ANO Construction Time and Verification
Time (Second), *KDDCUP* Dataset

(a) Completeness Verification		
(α, a)	AO Construction	Completeness Verification
(0.9, 0.75)	0.1649216111	0.0287629264
(0.9, 0.8)	0.1497754622	0.0305521828
(0.9, 0.85)	0.1478960015	0.032380496
(0.9, 0.9)	0.1486778035	0.0343804893
(0.9, 0.95)	0.1472718158	0.0411139978
Deterministic approach	15.341	N/A
(b) Correctness Verification		
(β, b)	ANO Construction	Correctness Verification
(0.9, 0.75)	0.031591748	0.4989598102
(0.9, 0.8)	0.0283305431	0.4090301879
(0.9, 0.85)	0.0284852679	0.3174848479
(0.9, 0.9)	0.0286167844	0.2299931051
(0.9, 0.95)	0.0287029023	0.1376942443
Deterministic approach	15.341	42,384

Third, we measured the performance of outlier recovery at the client side. In particular, we count the number of records whose outlieriness needs to be examined against the dataset D by the client, and report the result in Tables 4 a & 4 b. Both tables show that the number of records whose outlieriness needs to be examined is very small compared with the size of the dataset. For example, at most 8 records in *Letter* dataset (0.04 percent of the dataset) and at most 167 records in *KDDCUP* dataset (0.16 percent of dataset) are examined. Another interesting observation is that even though it is possible that the records that need to be examined include both true and false positive outliers, in our experiments, all of them are false positive outliers (original non-outliers). We also measured the time of outlier recovery at the client side. Fig. 6 shows the detailed result. Specifically, it is very efficient to eliminate the false positive outliers, due to the small number of records that need to be examined. For example, it needs no more than 0.1 second on the *Letter* dataset.

6.3 Overhead at Server Side

We measured the mining overhead at the server side as $|T_{D^+} - T_D|/T_D$, where T_D and T_{D^+} are the time of mining outliers from the original database D and the dataset $D^+ = D \cup \Delta D$. Figs. 7a and 7b show the mining overhead of the *Letter* dataset and *KDDCUP* dataset respectively. We observed that the insertion of artificial records (ARs) does not introduce significant mining overhead. For example, the mining overhead is no more than 1.2 percent for the *Letter* dataset, and at most 0.25 percent for the *KDDCUP* dataset. The overhead on *KDDCUP* dataset is much smaller because we insert the same number of artificial records for the same α, β, a, b values into a larger dataset. This proves that our approach is more suitable for datasets of large size.

7 RELATED WORK

Verifiable Computations for General-Purpose Computations. Goldwasser et al. [18] use interactive proofs to verify

TABLE 4
Number of Records Whose Outlieriness are
Examined During Post-Processing

a, b	α, β 0.75	α, β 0.8	α, β 0.85	α, β 0.9	α, β 0.95
(a) <i>Letter</i> Dataset (20K records)					
0.75	1	1	1	2	2
0.80	1	1	2	2	4
0.85	2	2	4	4	5
0.90	4	4	5	5	6
0.95	5	6	6	8	8
(b) <i>KDDCUP</i> Dataset (100K records)					
0.75	2	4	4	6	13
0.80	4	4	6	13	14
0.85	6	8	14	17	23
0.90	14	19	23	23	48
0.95	47	70	77	101	167

tractable computation. The proofs enable the client to verify the result's correctness in nearly-linear time. Gennaro et al. [17] formally define verifiable computing as a technique that enables a computationally weak client to verify the result correctness of outsourced computation. They propose a non-interactive proof construction approach. The proof enables the client to verify the result with complexity polynomial to the result's size. However, it demands an expensive pre-processing and input preparation phase to generate auxiliary information for verification. Setty et al. [44] build a general-purpose verification approach to support a wide range of computation including floating-point fractions and inequality comparisons. Parno et al. [40] propose to construct a VC scheme with public delegation and public verifiability from any attribute-based encryption scheme. PEPPER was proposed in [45]. It dramatically reduces the verification cost by using an argument system in which the verifier queries the linear probabilistically checkable proofs in an inexpensive way. PEPPER also reduces the prover's overhead so that its total work is not significantly more than the cost of executing the computation. [39] is the first general-purpose verification framework to demonstrate that the verification can be cheaper than the native computation. While it is reasonable in some scenarios that the same computation is executed on many different inputs, this is not ideal for the DMaS paradigm where the client outsources the data mining computations with a single input dataset.

Authenticated Outsourced Computations. Quite a few studies [1], [23], [38], [53] focus on the authentication of specific computations. Zhang et al. [53] define a new type of accumulation values so that the outsourced sum operation can be authenticated with two group elements and in constant time. Papadopoulos et al. [38] combine the accumulation values with the suffix tree to facilitate the authentication of outsourced pattern matching. The proof size is proved to be optimal as it includes at most ten accumulation values. Meanwhile, the proof can be readily constructed as the components have been pre-computed in the setup phase. Li et al. [25] design a novel authenticated indexing structure based on Merkle tree to detect the misbehavior by the cloud broker in the service selection process. Abadi et al. [1]

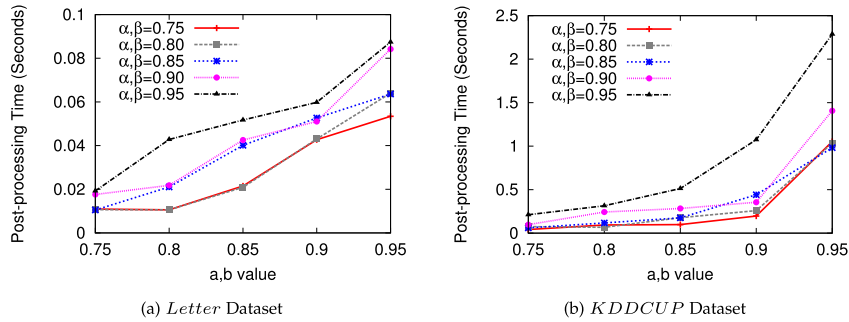


Fig. 6. Post-processing time.

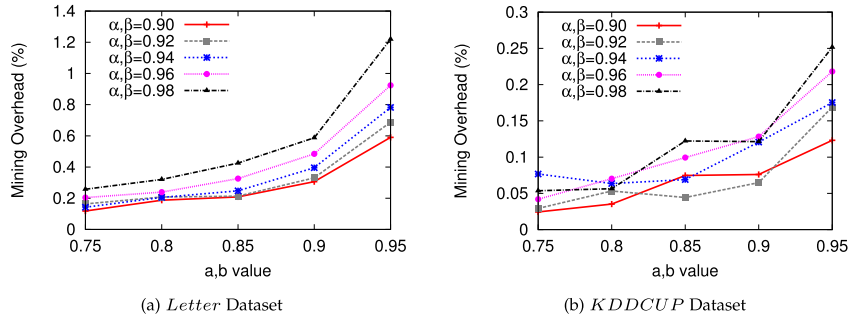


Fig. 7. Mining overhead.

propose a protocol that supports verifiable delegated private set intersection on outsourced datasets. The verification cost is dependent on the result size. In the crowdsourcing setting, Kupcu [23] combines cryptography and game theory to incentivize the rational workers to perform the computation correctly, and guarantee the result quality even in the existence of irrational workers who intentionally submit incorrect result.

Authenticated Query Evaluation in Data-Mining-as-a-Service Paradigm. Our problem falls into the category of integrity assurance of the Data-Mining-as-a-Service paradigm. Wong et al. [50] propose auditing techniques for outsourcing frequent itemset mining to an untrusted party. They generate a (small) artificial database such that all itemsets in the database are guaranteed to be frequent and their exact support counts are known. By hosting the artificial database with the original one and checking whether the server has returned all artificial itemsets, the data owner can verify whether the server has returned correct and complete frequent itemsets. However, the artificial itemsets can be easily identified by the server with publicly-available information about the original dataset. In order to fix the issue, Dong et al. [12] provide an approach to construct the artificial itemsets in the original dataset. These artificial itemsets are indistinguishable from the original ones, and guarantee to catch the server's dishonest result with high probability. Dong et al. [11] extend the work by proposing a verification framework based on cryptographic proofs to catch any incorrect mining result with 100 percent certainty. Other work [27] solves the result integrity verification problem for various computations such as Bayesian network learning and clustering. Their techniques on computations which are dramatically different from outlier mining and thus cannot be directly applied to our problem.

Authenticated Query Evaluation in Database-as-a-Service (DaS) Paradigm. The issue of integrity assurance for database management was initially raised in the Database-as-a-Service paradigm [20]. The focus is to assure the integrity of SQL query evaluation over the hosted relational databases. A few techniques have been proposed to provide assurance for SQL query evaluation [14], [20], [24], [46], [51]. For example, Hacigümüş et al. [20] propose a solution that uses conventional encryption techniques and additionally assign a bucket id to each attribute value to facilitate efficient evaluation. that partially executing query on the provider side, they can support range searches and joins in addition to exact match queries. The proposed solutions include Merkle hash trees [24], signatures on a chain of records [34], challenge tokens [46], and counterfeit records [51]. The concept of *verifiable database (VDB)* was first proposed in [7]. It uses the verifiable computation mechanisms to authenticate the query results on outsourced databases. Chen et al. [9], [10] design a new VDB framework to defend against the *forward automatic update* attack and support efficient incremental data updates. These work share the same result integrity concerns in the outsourcing paradigm. However, their focus is mainly SQL query evaluation, which is dramatically different from ours.

Data and Pattern Security of Data Mining. The problem of how to protect sensitive data and data mining results for the DMaS paradigm has caught much attention recently. A few work [13], [15], [30], [31], [47], [49] have been done under this theme. Wong et al. [49] consider utilizing a one-to-n item mapping together with non-deterministic addition of cipher items to protect the identification of individual items in the scenario that frequent pattern mining task is outsourced. Unfortunately, this work has potential privacy flaws; Molloy et al. [30] show how privacy can be breached in the framework of [49]. Tai et al. [47] consider the same scenario and

proposed a database transformation scheme that is based on a notion of k -support anonymity. To achieve k -support anonymity, they introduced a pseudo taxonomy tree; the third party server will discover the generalized frequent itemsets instead. Dong et al. [13] design data encoding schemes that preserve high accuracy in data deduplication computation while provide provable privacy guarantee. Dong et al. [15] consider the privacy leakage where the server leverages *functional dependency* to initiate the inference attack. They proposed a heuristic algorithm to block the inference channel by encrypting a small amount of insensitive data cells. Although these works focus on other mining tasks, their encryption techniques can be applied to our work to provide further protection on data and mining results.

Integrity Incentive Games. Morgan [32] initiates the study by incentivizing heterogeneous individuals in a game to improve the provision of public good. A fixed-prize raffle game is proposed. The author discovers a unique equilibrium when the benefit function is quasi-quadratic. Loiseau et al. [28] extends the study by adjusting the players' actions. Vaidya et al. [48] propose an incentive compatible protocol in the outsourced collaborative filtering computation. However, [48] does not consider the fact even though the result passes the verification, the client still can not put total belief in the result. Pham et al. [41] design an optimal contract by setting appropriate rewards, penalty and verification rate to guarantee the result correctness. Moghaddam et al. [29] model the interaction between the service provider and the client as a dynamic game. The service provider sends either legitimate or compromised signal to the client. Under various circumstances, the desired payoff values are set in order to motivate the server to play honestly. Different from [29], the server's decision of whether to cheat on the mining result is static in our model.

8 CONCLUSION

In the outsourcing paradigm, it is extremely important for the client to check if the result returned by the untrusted service provider is correct. In this paper, we concentrate on the authentication of outsourced outlier mining. We consider three types of untrusted servers with different adversarial knowledge. We proposed a lightweight authentication framework by constructing a set of artificial records (*ARs*) to catch any unsound or incomplete result with high probability guarantee. In order to incentivize the server to discover the outliers faithfully, we design a strategic game where the server always chooses to behave honestly to get the best payoff. We demonstrated the efficiency and effectiveness of our approach via an extensive set of experiments.

In the future, we plan to explore the deterministic authentication approach to catch the cheating behavior of the server with deterministic guarantee. We will also examine how to design authentication techniques that support data updates.

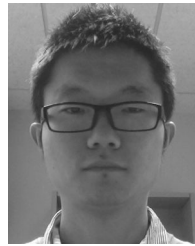
ACKNOWLEDGMENTS

This material is based upon work supported by the US National Science Foundation under Grant No. 1350324 and 1464800.

REFERENCES

- [1] A. Abadi, S. Terzis, and C. Dong, "VD-PSI: Verifiable delegated private set intersection on outsourced private datasets," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2016, pp. 149–168.
- [2] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," *ACM SIGMOD Rec.*, vol. 30, pp. 37–46, 2001.
- [3] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," in *Proc. Int. Workshop Secur. Protocols*, 1997, pp. 125–136.
- [4] F. Angiulli and F. Fassetto, "DOLPHIN: An efficient algorithm for mining distance-based outliers in very large datasets," *ACM Trans. Knowl. Discovery Data*, vol. 3, 2009, Art. no. 4.
- [5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, "Proof verification and the hardness of approximation problems," *J. ACM*, vol. 45, pp. 501–555, 1998.
- [6] V. Barnett and T. Lewis, *Outliers in Statistical Data*. New York, NY, USA: Wiley, 1994.
- [7] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Proc. Annu. Conf. Advances Cryptology*, 2011, pp. 111–131.
- [8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, pp. 93–104, 2000.
- [9] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 5, pp. 546–556, Sep./Oct. 2015.
- [10] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.
- [11] B. Dong, R. Liu, and H. W. Wang, "Integrity verification of outsourced frequent itemset mining with deterministic guarantee," in *Proc. Int. Conf. Data Mining*, 2013, pp. 1025–1030.
- [12] B. Dong, R. Liu, and H. W. Wang, "Result integrity verification of outsourced frequent itemset mining," in *Proc. Annu. Conf. Data Appl. Secur. Privacy*, 2013, pp. 258–265.
- [13] B. Dong, R. Liu, and H. W. Wang, "PraDa: Privacy-preserving data-deduplication-as-a-service," in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2014, pp. 1559–1568.
- [14] B. Dong and H. W. Wang, "ARM: Authenticated approximate record matching for outsourced databases," in *Proc. Int. Conf. Inf. Reuse Integr.*, 2016, pp. 591–600.
- [15] B. Dong, H. W. Wang, and J. Yang, "Secure data outsourcing with adversarial data dependency constraints," in *Proc. Int. Conf. Big Data Secur. Cloud*, 2016, pp. 73–78.
- [16] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, and A. van Moorsel, "Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2017.
- [17] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Annu. Conf. Advances Cryptology*, 2010, pp. 465–482.
- [18] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," in *Proc. Symp. Theory Comput.*, 2008, pp. 113–122.
- [19] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, pp. 186–208, 1989.
- [20] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. Int. Conf. Manage. Data*, 2002, pp. 216–227.
- [21] E. M. Knox and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets," in *Proc. Int. Conf. Very Large Data Bases*, 1998, pp. 392–403.
- [22] G. Kollios, D. Gunupulos, N. Koudas, and S. Berchtold, "An efficient approximation scheme for data mining tasks," in *Proc. Int. Conf. Data Eng.*, 2001, pp. 453–462.
- [23] A. Kupcu, "Incentivized outsourced computation resistant to malicious contractors," *IEEE Trans. Depend. Secure Comput.*, vol. PP, no. 99, p. 1, Nov. 2015.
- [24] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, "Dynamic authenticated index structures for outsourced databases," in *Proc. Int. Conf. Manage. Data*, 2006, pp. 121–132.
- [25] J. Li, A. Squicciarini, D. Lin, S. Sundareswaran, and C. Jia, "MMB^{cloud}-tree: Authenticated index for verifiable cloud service selection," *IEEE Trans. Depend. Secure Comput.*, vol. 14, no. 2, pp. 185–198, Mar./Apr. 2017.

- [26] R. Liu, H. W. Wang, A. Monreale, D. Pedreschi, F. Giannotti, and W. Guo, "AUDIO: An integrity auditing framework of outlier-mining-as-a-service systems," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2012, pp. 1–18.
- [27] R. Liu, H. W. Wang, and C. Yuan, "Result integrity verification of outsourced Bayesian network structure learning," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 713–721.
- [28] P. Loiseau, G. Schwartz, J. Musacchio, S. Amin, and S. S. Sastry, "Congestion pricing using a raffle-based scheme," in *Proc. Int. Conf. Netw. Games Control Optimization*, 2011, pp. 1–8.
- [29] M. M. Moghaddam, M. H. Manshaei, and Q. Zhu, "To trust or not: A security signaling game between service provider and client," in *Proc. Int. Conf. Decision Game Theory Secur.*, 2015, pp. 322–333.
- [30] I. Molloy, N. Li, and T. Li, "On the (In) security and (Im) practicality of outsourcing precise association rule mining," in *Proc. Int. Conf. Data Mining*, 2009, pp. 872–877.
- [31] A. Monreale and H. W. Wang, "Privacy-preserved data mining in cloud," in *Proc. Int. Workshop Secure Identity Manage. Cloud Environ.*, 2016, pp. 583–588.
- [32] J. Morgan, "Financing public goods by means of lotteries," *Rev. Econ. Stud.*, vol. 67, no. 4, pp. 761–784, 2000.
- [33] P. Morris, *Introduction to Game Theory*. Berlin, Germany: Springer, 2012.
- [34] M. Narasimha and G. Tsudik, "DSAC: Integrity for outsourced databases with signature aggregation and chaining," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2005, pp. 235–236.
- [35] H. V. Nguyen, V. Gopalkrishnan, and I. Assent, "An unbiased distance-based outlier detection approach for high-dimensional data," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2011, pp. 138–152.
- [36] M. J. Osborne, *An Introduction to Game Theory*. New York, NY, USA: Oxford Univ. Press, 2004.
- [37] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "LOCI: Fast outlier detection using the local correlation integral," in *Proc. Int. Conf. Data Eng.*, 2003, pp. 315–326.
- [38] D. Papadopoulos, C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Practical authenticated pattern matching with optimal proof size," in *Proc. VLDB Endowment*, vol. 8, pp. 750–761, 2015.
- [39] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 238–252.
- [40] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proc. Int. Conf. Theory Cryptography*, 2012, pp. 422–439.
- [41] V. Pham, M. H. R. Khouzani, and C. Cid, "Optimal contracts for outsourced computation," in *Proc. Int. Conf. Decision Game Theory Secur.*, 2014, pp. 79–98.
- [42] L. Qiu, Y. Li, and X. Wu, "Protecting business intelligence and customer privacy while outsourcing data mining tasks," *Knowl. Inf. Syst.*, vol. 17, pp. 99–120, 2008.
- [43] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Rec.*, vol. 29, pp. 427–438, 2000.
- [44] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish, "Taking proof-based verified computation a few steps closer to practicality," in *Proc. USENIX Conf. Secur. Symp.*, 2012, p. 12.
- [45] S. T. V. Setty, R. McPherson, A. J. Blumberg, and M. Walfish, "Making argument systems for outsourced computation practical (sometimes)," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2012, pp. 17–36.
- [46] R. Sion, "Query execution assurance for outsourced databases," in *Proc. Int. Conf. Very Large Data Bases*, 2005, pp. 601–612.
- [47] C.-H. Tai, P. S. Yu, and M.-S. Chen, "k-Support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 473–482.
- [48] J. Vaidya, I. Yakut, and A. Basu, "Efficient integrity verification for outsourced collaborative filtering," in *Proc. Int. Conf. Data Mining*, 2014, pp. 560–569.
- [49] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, "Security in outsourcing of association rule mining," in *Proc. Int. Conf. Very Large Data Bases*, 2007, pp. 111–122.
- [50] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, "An audit environment for outsourcing of frequent itemset mining," in *Proc. VLDB Endowment*, vol. 2, pp. 1162–1173, 2009.
- [51] M. Xie, H. Wang, J. Yin, and X. Meng, "Integrity auditing of outsourced data," in *Proc. Int. Conf. Very Large Data Bases*, 2007, pp. 782–793.
- [52] Y. Yang, D. Papadias, S. Papadopoulos, and P. Kalnis, "Authenticated join processing in outsourced databases," in *Proc. Int. Conf. Manage. Data*, 2009, pp. 5–18.
- [53] Y. Zhang, J. Katz, and C. Papamanthou, "IntegriDB: Verifiable SQL for outsourced databases," in *Proc. Conf. Comput. Commun. Secur.*, 2015, pp. 1480–1491.



Boxiang Dong received the PhD degree in computer science from Stevens Institute of Technology, New Jersey. He is an assistant professor in the Computer Science Department, Montclair State University, New Jersey. He is dedicated to facilitating the integration of big data analysis and cybersecurity. His research interests include verifiable computing, data mining, anomaly detection, data security, and privacy.



Hui Wang received the PhD degree in computer science from the University of British Columbia, Vancouver, Canada. She is an associate professor in the Computer Science Department, Stevens Institute of Technology, New Jersey. Her research interests include data management, data mining, database security, and data privacy. She is a member of the editorial boards of the *Journal of Information Technology and Architecture* and the *Journal of Information Systems*.



Anna Monreale received the MS and PhD degrees in computer science from the University of Pisa, in 2007 and 2011. She is a post-doc in the Computer Science Department, Pisa University and a member of the KDD-LAB, a joint research group with the Information Science and Technology Institute of the National Research Council in Pisa. Her research interests include privacy-aware data mining and data publishing and in privacy-preserving outsourcing of analytical tasks.



Dino Pedreschi is a full professor of computer science with the University of Pisa. His research interests include data mining, and in privacy-preserving data mining. He is a member of the PC of the main international conferences on data mining and knowledge discovery. He has been granted a Google Research Award (2009) for his research on privacy-preserving data mining and anonymity-preserving data publishing.



Fosca Giannotti is a senior researcher with the Information Science and Technology Institute of the National Research Council at Pisa, Italy, where she leads the KDD-LAB, a joint research initiative with the University of Pisa. Her research interests include data mining query languages, knowledge discovery support environment, web-mining, spatio-temporal data mining, and privacy preserving data mining. She has been the coordinator of various projects and she served in the scientific committee of various conferences.



Wenge Guo received the PhD degree in biostatistics from the University of Cincinnati, Ohio. He is an associate professor in the Department of Mathematical Sciences, New Jersey Institute of Technology. His research interests include large-scale multiple testing, high-dimensional data analysis, bioinformatics, and machine learning. He is a member of the editorial boards of the *Statistics and Probability Letter*, the *PLoS One*, and the *Calcutta Statistical Association Bulletin*.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Bilateral Privacy-Preserving Utility Maximization Protocol in Database-Driven Cognitive Radio Networks

Zhikun Zhang¹, Student Member, IEEE, Heng Zhang², Member, IEEE,
Shibo He¹, Member, IEEE, and Peng Cheng¹, Member, IEEE

Abstract—Database-driven cognitive radio has been well recognized as an efficient way to reduce interference between Primary Users (PUs) and Secondary Users (SUs). In database-driven cognitive radio, PUs and SUs must provide their locations to enable dynamic channel allocation, which raises location privacy breach concern. Previous studies only focus on unilateral privacy preservation, i.e., only PUs' or SUs' privacy is preserved. In this paper, we propose to protect bilateral location privacy of PUs and SUs. The main challenge lies in how to coordinate PUs and SUs to maximize their utilities provided that their location privacy is protected. We first introduce a quantitative method to calculate both PUs' and SUs' location privacy, and then design a novel privacy preserving Utility Maximization protocol (UMax). UMax allows for both PUs and SUs to adjust their privacy preserving levels and optimize transmit power iteratively to achieve the maximum utilities. Through extensive evaluations, we demonstrate that our proposed protocol can efficiently increase the utilities of both PUs and SUs while preserving their location privacy.

Index Terms—Location privacy, bilateral privacy preservation, cognitive radio networks

1 INTRODUCTION

COGNITIVE radio networks have been well recognized as an efficient way to increase the spectrum utilization and thus alleviate the spectrum scarcity issue [1], [2], [3]. In cognitive radio networks, there are two types of users: Primary Users (PUs) and Secondary Users (SUs). PUs have the priority to access the spectrum since they have registered a chunk of spectrum from the spectrum management entity such as FCC, whereas SUs are allowed to access PUs' channels only when the interested channels are vacant.

Cognitive radio networks have a wide spectrum of potential applications including smart grid networks, public safety networks, medical body area networks, etc [4]. Let's take medical body area networks (MBAN) as an example, MBAN is a promising way to allow body sensors to reliably and inexpensively collect the vital signs of patients and relay the monitoring information to clinicians for rapid diagnosing. Quality of service is a key requirement for MBAN, which requires clean and less crowded spectrum. However, the 2.4 GHz industrial, scientific and medical band is too crowded to support the life-critical medical

applications. By using some PUs' vacant band on a *secondary basis*, i.e., acting as an SU, quality of service for MBAN can be better guaranteed [4], [5], [6], [7].

To enable dynamic channel access, SUs should be aware of which channels are locally available for reuse. There are mainly two ways for achieving this: 1) spectrum sensing and 2) database querying. Spectrum sensing method requires SUs to be equipped with specific sensors to detect the locally available channels [8], [9], [10], [11], [12]. Interference may occur when the sensors output false results, which may be caused by obstruction or channel fading. Database querying method requires SUs to provide their accurate locations to a centralized database [13], [14], [15]. In such a way, SUs can facily figure out locally available channels and thus avoid interference by querying a database, which maintains an up-to-date spectrum availability repository.

Despite the huge advantage of database-driven cognitive radio networks, SUs' locations are exposed to enable efficient channel allocation, which may breach SUs' location privacy. For example, in the MBAN application, the location privacy of patients will inevitably be breached. Besides, the response from database contains information relevant to the distance between PUs and a queried SU, a malicious SU may infer PUs' locations through seemingly innocuous multiple database queries. The potential privacy breach risk of both PUs and SUs have been an obstacle to promote database-driven cognitive radio networks.

Previous studies mainly focus on unilateral location privacy preservation in database-driven cognitive radio networks, i.e., they assume that one party (PUs or SUs) is trustworthy and try to preserve the other's privacy [16], [17].

- Z. Zhang, S. He, and P. Cheng are with State Key Laboratory of Industrial Control Technology, Cyber Innovation Joint Research Center, Zhejiang University, Hangzhou 310000, China.
E-mail: {zhangzhk, s18he}@zju.edu.cn, pcheng@iipc.zju.edu.cn.
- H. Zhang is with Huaihai Institute of Technology, Lianyungang, Jiangsu 222005, China, and also with School of Computing, Engineering and Mathematics, Western Sydney University, Sydney, NSW 2150, Australia.
E-mail: ezhangheng@gmail.com.

Manuscript received 1 Nov. 2016; revised 23 Nov. 2017; accepted 24 Nov. 2017. Date of publication 8 Dec. 2017; date of current version 18 Mar. 2020.
(Corresponding author: Shibo He.)

Digital Object Identifier no. 10.1109/TDSC.2017.2781248

Furthermore, they fail to quantify the privacy preserving levels (PPLs) of PUs and SUs, making it difficult to analyze the tradeoff between both parties' PPLs and utilities. As aforementioned, both PUs' and SUs' location privacy could be potentially breached and thus should be preserved simultaneously. Simply applying previous mechanisms to preserve the privacy of PUs and SUs separately will suffer severe utility loss for both parties. As rational users, both parties intend to maximize their PPLs to efficiently thwart the attacker's threat. However, PPL and utility are always a paradox, in the sense that the increase of PPL will inevitably decrease SUs' available spectrum. Therefore, the bilateral location privacy issue should be jointly addressed, in which PUs and SUs can adjust their PPLs in order to maximize their utilities.

To address the above challenging issues, we first adopt the celebrated notion of differential privacy [18] to simultaneously preserve PUs' and SUs' privacy. Then, we design a quantitative privacy-preserving framework which is flexible for both PUs and SUs to adjust their PPLs. With this framework, we proceed to propose a novel privacy preserving Utility Maximization protocol (UMax) that allows both PUs and SUs to adjust their PPLs to achieve their maximum utilities. In UMax, PUs and SUs exchange information to decide their optimal PPLs. First, SUs decide their optimal PPLs according to their utility function, which incorporate SUs' revenue and privacy lost. At the query epoch, SUs send their interested channels and expected transmit radius to database together with obfuscated locations generated based on their optimal PPLs. Second, based on SUs' expected information, the database decides PUs' optimal PPLs and SUs' available transmit power to maximize PUs' utilities.

The main contributions of this paper are summarized as follows:

- (1) To the best of our knowledge, this is the first work that simultaneously considers location privacy preservation for both PUs and SUs. Drawing on the concept of differential privacy, a quantitative framework is proposed to simultaneously preserve both PUs' and SUs' location privacy.
- (2) Based on the quantitative framework, we propose a novel database access protocol UMax, which allows both PUs and SUs to adjust their PPLs to maximize their utilities. We further prove the existence of optimal PPLs for both PUs and SUs.
- (3) We further generalize UMax protocol to the scenario where there are multiple PUs and multiple SUs with complex relative location combinations and allocation strategies.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the basic database access protocol and threat model. In Section 4, we propose novel location privacy-preserving mechanisms for both PUs and SUs. In Section 5, considering the single PU and single SU situation, we develop a new database access protocol that adjusts PU's and SU's PPL to improve their utility, respectively. Then we extend the proposed protocol to the multiple PUs multiple SUs situation in Section 6. Extensive simulations are performed in Section 7 to demonstrate the performance of the proposed protocol. Finally, Section 8 concludes this paper.

The notations that will be used in this paper are summarized in Table 1.

TABLE 1
Notations

$r_{p,i}^0$	PU_i 's protected contour radius
$r_{p,i}^\epsilon$	Length added to PU_i 's protected contour radius
$L_{p,i}$	PU_i 's PPL, i.e., $E(r_{p,i}^\epsilon)$
$T_{p,i}$	PU_i 's revenue
$C_{p,i}^{pri}$	PU_i 's privacy cost
loc_j	SU_j 's accurate location
loc_j'	SU_j 's randomized location
$r_{s,j}^0$	SU_j 's required transmit radius
R_j	SU_j 's maximum transmit radius
P_j	SU_j 's maximum transmit power
$r_{s,j}^\epsilon$	SU_j 's privacy-preserving circle radius
$L_{s,j}$	SU_j 's PPL, i.e., $r_{s,j}^\epsilon$
$T_{s,j}$	SU_j 's revenue
$C_{s,j}^{pri}$	SU_j 's privacy cost
$C_{s,j}^{buy}$	SU_j 's cost to buy spectrum

2 RELATED WORK

Most previous studies on cognitive radio focus on the performance improvements of spectrum sensing [19], [20], [21], [22] or security issues [23], [24], whereas privacy issues are not well been studied, especially in the area of database driven cognitive radio. Although there are little works concern about privacy preservation in cognitive radio, plenty of privacy enhancing techniques are studied in other applications, which provide us useful reference on the design of privacy preserving technique for cognitive radio networks. We can classify the state-of-the-art privacy enhancing techniques into three categories: anonymization technique, cryptographic technique and differential privacy.

k -anonymity is the most widely used anonymization technique. One approach to achieve k -anonymity is to use *dummy locations* [25]. This technique properly selects $k - 1$ dummy points, and then performs k queries to database together with the real location. Another efficient method is *cloaking* [26], [27], which creates a dummy region that involves k different points sharing the same property, and then queries the database with the dummy region. l -diversity [28] and t -closeness [29] are two anonymization techniques proposed to address the weaknesses of k -anonymity when homogeneity exists in the sensitive values in a group. However, the intrinsic drawback of k -anonymity is that a mechanism is difficult to be proved to satisfy this notion, since the attacker's auxiliary information may violate the guarantee of k -anonymity. In addition, k -anonymity based approaches are difficult to quantify the privacy preserving level.

Cryptography is another location privacy-preserving technique, which has been widely used [30], [31], [32]. This technique transforms all the data in a query process to a different space. The query result can be mapped back to spatial information only by the user. However, the computational overhead of cryptography based technique is too high.

The notion of differential privacy [18] comes from the area of statistical database. Its goal is to preserve individual's privacy while achieving good statistical accuracy. Recently, Andres et. al. [33] proposed the ϵ -geo-indistinguishability mechanism, which generalized the notion of differential

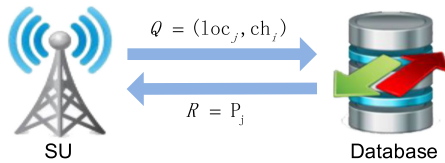


Fig. 1. Basic database access protocol.

privacy to location based service. The main advantage of differential privacy is that the privacy guarantee is independent of attacker's auxiliary information, i.e., the mechanism has no need to update when new types of attack emerges. More importantly, differential privacy provides a solid mathematical definition which is convenient to quantify the privacy preserving level [34], [35]. Due to the advantage of differential privacy, we adopt the notion of differential privacy to preserve the location privacy for both PUs and SUs.

The existing works on the location privacy in database-driven cognitive radio concern only about unilateral privacy preservation. In [16], Gao et al. proposed a cryptography based location privacy-preserving protocol called PSAIR for SUs. PSAIR allows for SUs to access the locally available channels and preserve location privacy simultaneously. Bahrak et al. [17] pointed out that a malicious SU can infer PU's location through seemingly innocuous database queries. Then they proposed a k -anonymity based mechanism to preserve PU's privacy. To the best of our knowledge, there is no existing work that addresses the bilateral privacy preservation for PUs and SUs simultaneously.

3 BACKGROUND AND THREAT MODEL

In this section, we first introduce the basic database access protocol that do not consider the privacy issues. Then, we present the threat model for the basic database access protocol as well as the problems to be addressed in this paper.

3.1 Basic Database Access Protocol

A typical database-driven cognitive radio network comprises three main components: PUs, SUs and spectrum management database. The database maintains PUs' locations and spectrum utilization informations. Whenever PUs change their spectrum utilization informations, they will notify database to update the repository.

We assume there are M PUs and N SUs in the network, which are denoted by PU_i and SU_j , respectively. Each PU_i owns a specific channel ch_i ,¹ which has two states, i.e., occupied or vacant. When ch_i is occupied, PU_i will delimit a protected contour where no SUs can transmit, and if an SU is beyond PU_i 's protected contour, it is allowed to transmit with a certain power. Intuitively, the farther an SU is located from PU_i , the larger power it can transmit. When ch_i is vacant, it can be accessed by SUs freely.

A basic database access protocol without considering the privacy issues is shown in Fig. 1. The database query process is conducted in the beginning of each time slot, within which the channel access state do not change. At the beginning of each time slot, SUs send their queries $Q = (loc_j, ch_j)$ to the database, where $loc_j = (x_j, y_j)$ is the accurate location of SU_j , and ch_j is the channel with the best quality for SU_j . Then, the

database returns $R = P_j$ to SU_j , where P_j is the maximum transmission power (MTP) for SU_j when it access ch_j . The MTP can be calculated by the following function [17]:

$$P_j = \begin{cases} 0, & d_{ij} \leq r_{p,i}^0, \\ h(d_{ij} - r_{p,i}^0), & d_{ij} > r_{p,i}^0, \end{cases} \quad (1)$$

where $r_{p,i}^0$ is the protected contour radius of ch_i , d_{ij} is the distance between SU_j and PU_i , and $h(\cdot)$ is a continuous monotone increasing function.

3.2 Threat Model and Assumptions

In the basic database access protocol, both database and SUs are assumed to be trustworthy. However, this assumption may not always guaranteed in reality. The reason is that a curious database manager may collect SUs' location information, which should be reported to the database to achieve the locally available channels, to make market decision or sales strategy. On another hand, a malicious SU may infer PUs' locations through multiple seemingly innocuous queries as depicted in Section 4.

We assume that the database is an affiliated entity of PUs, e.g., China Mobile would maintain a spectrum management database if it decides to share the registered spectrum with SUs. Thus, we further assume that PUs and database trust each other, and database will not breach PUs' operational information. Besides, we assume that a sophisticated malicious SU can obtain the MTP function which the database adopts [17].

3.3 Problem of Interest

To defend the potential privacy threat, both PUs and SUs need to adopt an quantitative mechanism to preserve their location privacy. However, unrestricted increase of PPL may seriously decrease both PUs' and SUs' utility. Thus, we need to deal with the following two problems in our paper:

- (1) How to devise quantitative mechanisms to preserve the location privacy for both PUs and SUs?
- (2) How to design an efficient database access protocol which allows for both PUs and SUs to adjust their PPLs to achieve the maximum utilities?

4 QUANTITATIVE PRIVACY-PRESERVING MECHANISM

In this section, we first introduce the quantitative privacy-preserving mechanisms for both PUs and SUs. Then, an interference free framework is proposed to facilitate the analysis of the bilateral privacy-preserving protocol.

4.1 Quantitative Privacy-Preserving Mechanism for PUs

First, we present the location privacy threat for PUs when they do not adopt any privacy-preserving mechanisms. In brief, a malicious SU can infer PU's location through multiple seemingly innocuous queries as Fig. 2a shows. Specifically, one sophisticated malicious SU, which obtained the MTP function of database, can compute its maximum transmit radius, i.e., d_1 , d_2 , d_3 and d_4 , in each query location. Thus, after multiple queries in different locations, the malicious SU can choose at least four query results, that contain

1. We will use PU_i and ch_i interchangeably in the following part.

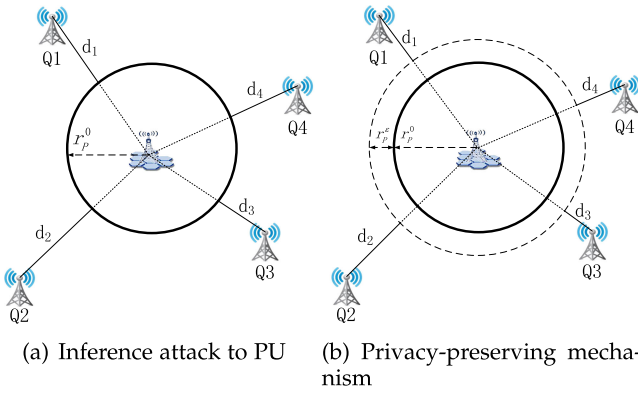


Fig. 2. PU's location privacy threat and countermeasure. Q_1 , Q_2 and Q_3 stand for three different query locations, d_1 , d_2 , d_3 and d_4 stand for their corresponding maximum transmit radius.

available transmit power, to locate PU by solving the following equations set

$$\begin{cases} (x_1 - x_p)^2 + (y_1 - y_p)^2 = (d_1 + r_p^0)^2 \\ (x_2 - x_p)^2 + (y_2 - y_p)^2 = (d_2 + r_p^0)^2 \\ (x_3 - x_p)^2 + (y_3 - y_p)^2 = (d_3 + r_p^0)^2 \\ (x_4 - x_p)^2 + (y_4 - y_p)^2 = (d_4 + r_p^0)^2, \end{cases} \quad (2)$$

where (x_i, y_i) is the location of Q_i , (x_p, y_p) is the location of the PU to be inferred, r_p^0 is the protected contour radius of the PU. With the above equations set, the malicious SU can uniquely determine PU's accurate location (x_p, y_p) and the corresponding protected contour radius r_p^0 .

To thwart malicious SU's inference attack, we propose an obfuscation based mechanism as Fig. 2b shows. Before computing SUs' MTP, the database will add a random length to PUs' real protected contour radius, i.e., $r_p = r_p^0 + r_p^\epsilon$. With the randomized MTP, the r_p^0 term in equations set 2 is different, making the calculation of PU's accurate location (x_p, y_p) difficult. In this paper, we adopt exponential distribution to generate the required random distance r_p^ϵ . The corresponding probability density function is given by

$$g(r_p^\epsilon) = \begin{cases} \frac{1}{b} e^{-\frac{r_p^\epsilon}{b}}, & r_p^\epsilon > 0, \\ 0, & r_p^\epsilon \leq 0, \end{cases}$$

where b is the rate parameter. Notice that r_p^ϵ is always positive, since otherwise the obfuscated protected contour will be less than the required one which may cause interference.

Intuitively, larger noise means higher PPL, so that we can adopt the expectation of r_p^ϵ , i.e., $E[r_p^\epsilon]$, to denote PU's PPL L_p .²

4.2 Quantitative Privacy-Preserving Mechanism for SUs

Recall that SUs should report their accurate locations to achieve the locally available channels, leading to serious privacy breach threat. Therefore, drawing on the privacy-preserving mechanism in [33], we propose l -geo-indistinguishability (l -geoin) mechanism to preserve SUs' location privacy.

Informally, l -geoin allows for SUs to report randomized locations generated by certain distribution to the database.

2. We will use $E[r_p^\epsilon]$ and L_p interchangeably to denote PU's PPL.

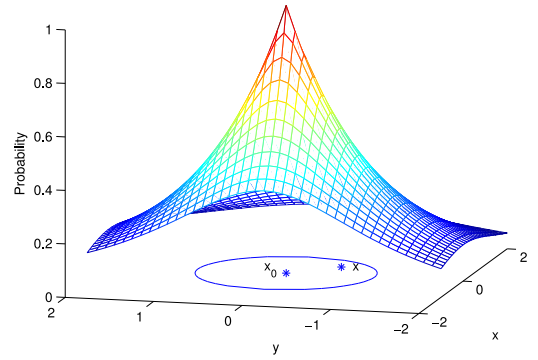


Fig. 3. SU located in x_0 generate randomized location x with probability as shown in "Probability" axis.

l -geoin guarantees that after receiving SUs' randomized locations, a curious database manager cannot figure out SUs' accurate locations with high confidence. The formal definition of l -geoin is given as follows:

Definition 1. A privacy-preserving random mechanism satisfies l -geoin if and only if for a reported location x , we have

$$\frac{P(x|x_0)}{P(x|x'_0)} \leq e^l, \forall r_s^\epsilon > 0, d(x_0, x'_0) \leq r_s^\epsilon, \\ l = \epsilon r_s^\epsilon,$$

where r_s^ϵ is the radius of the largest area where SU's privacy-preserving location may lie through the random mechanism, x_0 and x'_0 are two accurate locations of SU that may report x , $d(x_0, x'_0)$ is the distance between x_0 and x'_0 .

Definition 1 shows that whether SU's accurate location is x_0 or x'_0 , the reported location can be x with certain probability, and their probability difference is upper bounded by e^l if the distance between x_0 and x'_0 is less than r_s^ϵ . Notice that a larger l means more difficult to infer SU's accurate location, leading to higher PPL.

In [33], the authors consider that for a given radius r_s^ϵ , a smaller ϵ means higher PPL, they name it ϵ -geo-indistinguishability (ϵ -geoin) mechanism. From our perspective, given a certain l , we can adjust ϵ to achieve a larger r_s^ϵ , every expected r_s^ϵ corresponds to an ϵ . Thus, we can use r_s^ϵ to denote SU's PPL L_s ,³ where a larger r_s^ϵ means that SU can obfuscate its location in a larger scale with l -geoin mechanism. By this notion, SU can choose its protected scale flexibly, rather than adjust its PPL in a fixed scale as ϵ -geoin mechanism.

Andres et al. [33] proved that the two dimensional Laplacian distribution satisfies ϵ -geoin, and thus satisfies l -geoin. The probability density function of two dimensional Laplacian distribution is given by

$$f(x|x_0) = \frac{\epsilon^2}{2\pi} e^{-\epsilon d(x_0, x)}, \quad (3)$$

where $x_0 \in \mathbb{R}^2$ is the accurate location of SU and $x \in \mathbb{R}^2$ is the corresponding randomized location, $d(x_0, x)$ is the distance between x_0 and x . Fig. 3 shows how this mechanism works. An SU located at x_0 can generate a randomized

3. We will use r_s^ϵ and L_s interchangeably to denote SU's PPL.

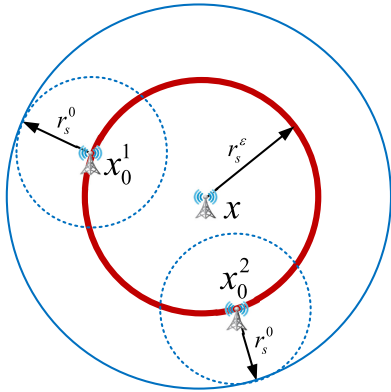


Fig. 4. Interference free framework. x is SU's reported randomized location, x_0^1 and x_0^2 are two possible accurate location of the SU.

location x within a circle with r_s^e radius, i.e., the required PPL, from two dimensional Laplacian distribution.

4.3 Interference Free Framework

In this section, we propose an interference free framework to facilitate the decision procedure of database, as Fig. 4 shows.

We assume each SU has a required transmission radius r_s^0 based on its service requirement, and the database decide each SU's MTP based on its reported location and required transmit radius. In each query process, each SU would report a randomized location x to the database to preserve its location privacy, such that a curious database manager cannot distinguish the SU's accurate location within a circle of radius r_s^e , as the inner solid circle shows in Fig. 4. From database's perspective, when allocating spectrum to SUs, all SUs' outer circles should not intersect with each other to avoid interference. The reason is that the furthest distance SU may transmit is the boundary of the outer solid circle, since the accurate location of SU maybe in the boundary of the inner solid circle, as x_0^1 and x_0^2 shows.

For brevity, we use *privacy preserving circle* and *interference free circle* to describe SUs' requirement in the following part, where privacy preserving circle and interference free circle correspond to the inner solid circle and outer solid circle in Fig. 4, respectively. Notice that, when any two SUs' privacy preserving circles intersect with each other, the possible location between the two SUs can be arbitrary close from database' perspective. Namely, no matter how much transmit radius we allocate to the two SUs, they may interfere with each other. Thus, only one SU is allowed to transmit.

5 PRIVACY-PRESERVING DATABASE ACCESS PROTOCOL

In this section, we propose a new database access protocol UMax to allow for PUs and SUs to choose the optimal PPLs to achieve their maximum utilities. The single PU single SU situation is considered in this section, and we will generalize the proposed protocol to multiple PUs multiple SUs situation in Section 6.

We first provide an overview to the proposed UMax protocol in Section 5.1, then deeply study the optimal decision of SU and PU in Sections 5.2 and 5.3.

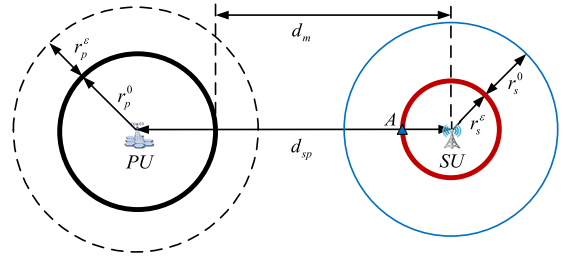


Fig. 5. Relative location between PU and SU.

5.1 Overview to UMax Protocol

In this section, we provide an overview to the proposed database access protocol UMax.

Step 1: SU query the database with its optimal PPL. SU with accurate location (x, y) query the database with $Q = (ch_i, r_s^0, loc', r_s^e)$, where ch_i is SU's interested channel, and r_s^0 is SU's expected transmit radius based on its service requirement. $loc' = (x', y')$ is a randomized location generated with the optimal PPL r_s^e , which can be calculated by solving the following optimization problem:

Problem 5.1.

$$\begin{aligned} \arg \max_{L_s} U_s(L_s) &= T_s - C_s^{buy} - C_s^{pri}, \\ s.t. \quad L_s &> 0. \end{aligned}$$

In Problem 5.1, $U_s(L_s)$ is SU's utility function which consists of three parts: T_s is SU's revenue by utilizing the spectrum, C_s^{buy} is SU's payment to PU for utilizing the spectrum, and C_s^{pri} is SU's privacy cost. Details of SU's optimal decision will be presented in Section 5.2.

Step 2: Database decides SU's MTP based on the channel state. When SU's interested channel is vacant, database calculates SU's MTP based on r_s^0 . When the channel is occupied, database should first decide PU's optimal PPL then calculate SU's MTP. PU's optimal PPL can be calculated by solving the following optimization problem:

Problem 5.2.

$$\begin{aligned} \arg \max_{L_p} E[U_p(L_p)] &= T_p - C_p^{pri}, \\ s.t. \quad L_p &> 0. \end{aligned}$$

In Problem 5.2, $E[U_p(L_p)]$ is PU's utility function in the expectation sense which consists of two parts: T_p is PU's revenue by selling spectrum to SU which is equal to C_s^{buy} , and C_p^{pri} is PU's privacy cost. Details of PU's optimal decision will be presented in Section 5.3.

Then, the database can generate a random distance r_p^e based on the optimal PPL, i.e., $E[r_p^e]$. SU's maximum transmission radius (MTR)⁴ is given by $R = d_{sp} - r_p^0 - r_p^e - r_s^e$ as Fig. 5 shows, where d_{sp} is the distance between PU and SU.

5.2 SU's Optimal Decision

In this section, we elaborate the optimal decision of SU.

4. Notice that SU's MTP is decided by its maximum transmission radius so that we would use MTP and MTR interchangeably in the following part.

We define SU's revenue as

$$T_s = k_2\pi(r_s^0)^2, \quad (4)$$

where k_2 is SU's revenue of using unit area spectrum, and r_s^0 is SU's expected transmit radius.

As Fig. 4 shows, SU's expected transmit radius is r_s^0 . It seems that PU should charge SU based on r_s^0 . However, to satisfy SU's privacy-preserving requirement, PU need to allocate an area of radius $r_s^0 + r_s^\epsilon$ to avoid interference. To incentivize PU to share its spectrum, SU need to pay for the extra spectrum incurred by its privacy preserving requirement. Thus, we can define SU's payment as

$$C_s^{buy} = k_1\pi(r_s^\epsilon + r_s^0)^2,$$

where k_1 is the payment for using *unit area* spectrum. $\pi(r_s^\epsilon + r_s^0)^2$ is the area that PU allocate to SU for accessing a given spectrum. Since SU's PPL L_s is equal to r_s^ϵ , we can transform the above formula into the following form

$$C_s^{buy} = k_1\pi(L_s + r_s^0)^2. \quad (5)$$

The decrease of SU's PPL may increase its revenue with larger privacy breach risk. This kind of privacy breach risk is the cost which should be included into PU's utility function. Privacy cost can be defined as follows [36]

$$C_s^{pri} = \frac{k_s}{L_s}, \quad (6)$$

where k_s is the privacy cost coefficient.

Combining (4), (5) and (6), we can rewrite Problem 5.1 as

Problem 5.3.

$$\begin{aligned} \arg \max_{L_s} E[U_s] &= k_2\pi(r_s^0)^2 - k_1\pi(L_s + r_s^0)^2 - \frac{k_s}{L_s}, \\ \text{s.t. } L_s &> 0. \end{aligned}$$

Then, we show that Problem 5.3 has optimal solution and provide efficient method to achieve it.

Theorem 5.1. *There exists an optimal PPL for SU, i.e., L_s^* , to solve Problem 5.3.*

Proof. The second order derivative of $E[U_s]$ is given by

$$\frac{d^2 E[U_s]}{dL_s^2} = -2k_1\pi - 2\frac{k_s}{L_s^3},$$

Notice that $\frac{d^2 E[U_s]}{dL_s^2}$ is always less than zero, which means $E[U_s]$ is a concave function. In addition, the constraint of Problem 5.3 is convex. Thus, Problem 5.3 is a convex optimization problem, which have an optimal solution. \square

Since Problem 5.3 is a convex optimization problem, we can solve it by existing convex solver such as gradient decent method [37].

5.3 PU's Optimal Decision

In this section, we elaborate the optimal decision of PU.

To avoid interference, SU's expected transmit radius r_s^0 may not be fully satisfied. To incentivize SU to reuse the vacant spectrum continuously, PU can reduce the unit price

of spectrum to compensate expenses of SU. Thus, PU's revenue can be defined as

$$\begin{aligned} T_p &= \frac{R}{r_s^0} k_1\pi(r_s^\epsilon + R)^2 \\ &= \frac{k_1\pi}{r_s^0} (d_m - L_p - r_s^\epsilon)(d_m - L_p)^2, \end{aligned} \quad (7)$$

where R is SU's MTR, and $\frac{R}{r_s^0}$ stands for SU's satisfaction ratio.

Similar to the analysis of SU, we can define PU's privacy cost as follows

$$C_p^{pri} = \frac{k_p}{L_p} \quad (8)$$

Combining (7) and (8), we can rewrite PU's expected utility as

$$E[U_p] = \frac{k_1\pi}{r_s^0} (d_m - L_p - r_s^\epsilon)(d_m - L_p)^2 - \frac{k_p}{L_p}.$$

The optimal value of L_p lies in $[d_m - r_s^\epsilon - r_s^0, d_m - r_s^\epsilon]$ as Fig. 5 shows. The reason is that if $L_p \leq d_m - r_s^\epsilon - r_s^0$, SU's expected transmit radius can always be satisfied. In this case, the revenue $T_p = k_1\pi(r_s^\epsilon + r_s^0)^2$ is a constant, we can always achieve better utility by extending L_p . In addition, if $L_p \geq d_m - r_s^\epsilon$, the database can not decide SU's MTR. Since arbitrary transmit radius may cause interference with PU when SU is located in point A as Fig. 5 shows. Thus, Problem 5.2 becomes

Problem 5.4.

$$\begin{aligned} \arg \max_{L_p} E[U_p] &= \frac{k_1\pi}{r_s^0} (d_m - L_p - r_s^\epsilon)(d_m - L_p)^2 - \frac{k_p}{L_p}, \\ \text{s.t. } L_p &> 0, \\ d_m - r_s^\epsilon - r_s^0 &\leq L_p \leq d_m - r_s^\epsilon. \end{aligned}$$

Then, we show that Problem 5.4 has optimal solution and provide efficient method to achieve it.

Theorem 5.2. *There exists an optimal PPL for PU, i.e., L_p^* , to solve Problem 5.4.*

Proof. The first order derivative of $E[U_p]$ is given by

$$\begin{aligned} \frac{dE[U_p]}{dL_p} &= \frac{k_1\pi}{r_s^0} \left[-3L_p^2 + (6d_m - 2r_s^\epsilon)L_p \right. \\ &\quad \left. - 3d_m^2 + 2r_s^\epsilon d_m \right] + \frac{k_p}{L_p^2}. \end{aligned}$$

Recall that the roots of equation $\frac{dE[U_p]}{dL_p} = 0$ is the stationary points of the objective function $E[U_p]$. Notice that $\frac{dE[U_p]}{dL_p} = 0$ is a *quartic equation* and have at most 4 real roots, such that $E[U_p]$ have at most 4 stationary points. Further, we observe that $E[U_p]$ is a continuous function, and the feasible region of Problem 5.4 lies in a closed interval $L_p \in [d_m - r_s^\epsilon - r_s^0, d_m - r_s^\epsilon]$. Thus, Problem 5.4 has an optimal solution in the interval $[d_m - r_s^\epsilon - r_s^0, d_m - r_s^\epsilon]$. \square

To calculate the optimal PPL, we first calculate the real roots for equation $\frac{dE[U_p]}{dL_p} = 0$ with off the shelf quartic equation solver [38], and calculate the value of $E[U_p]$ using the real roots and the boundary value, i.e., $d_m - r_s^\epsilon - r_s^0$ and $d_m - r_s^\epsilon$. Then,

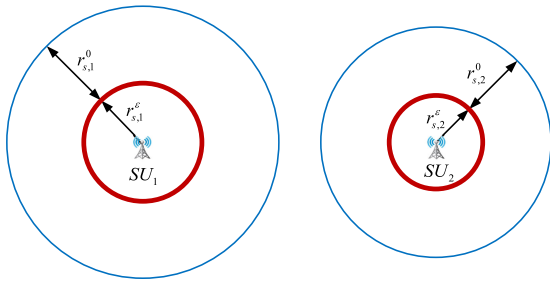


Fig. 6. Interference free circles do not intersect with each other.

we compare all the calculated value of $E[U_p]$ to choose the optimal PPL. If there exist two L_p at which $E[U_p]$ achieves the optimal value, we will choose the larger L_p as the optimal PPL.

6 MULTIPLE PUS MULTIPLE SUS SITUATION

In this section, we generalize UMax to multiple PUs multiple SUs situation. Recall that different PUs occupy different channels so that PUs can allocate their spectrum separately. Thus, we can reduce the multiple PUs multiple SUs problem to the single PU multiple SUs problem. In addition, when the number of SUs that apply for the same channel are arbitrary, the relative location combinations and allocation strategies is infinite, which makes it difficult to analyze the general case. To start, we consider the situation where there are at most two SUs applying for the same channel.

In practice, we extend UMax as follows:

Step 1. SUs decide their optimal PPLs based on Problem 5.3, and query the database with $Q = (ch_i, loc'_j, r_{s,j}^0, r_{s,j}^\epsilon)$, where ch_i is SU_j 's interested channel, loc'_j is the randomized location based on optimal PPL $r_{s,j}^\epsilon$, and $r_{s,j}^0$ is SU_j 's expected transmit radius.

Step 2. Database decide PU_i 's allocation strategy based on ch_i 's channel state. When ch_i is vacant, database decide each SU's optimal transmit radius directly on the condition that SUs do not interfere with each other. And when ch_i is occupied, database should first decide PU_i 's optimal PPL, and decide SU's optimal transmit radius on the condition that PU_i do not interfere with all SUs.

In the following section, we will discuss PU's allocation strategy in different channel states.

6.1 The Channel is Vacant

We define PU_i 's revenue as $T_{p,i} = T_{p,i}^1 + T_{p,i}^2$, and

$$T_{p,i}^1 = \frac{R_1}{r_{s,1}^0} k_1 \pi (r_{s,1}^\epsilon + R_1)^2,$$

$$T_{p,i}^2 = \frac{R_2}{r_{s,2}^0} k_1 \pi (r_{s,2}^\epsilon + R_2)^2,$$

where $T_{p,i}^1$ and $T_{p,i}^2$ are revenues from SU_1 and SU_2 , respectively, $R_j (j = 1, 2)$ is SU_j 's MTR, $r_{s,j}^\epsilon$ is SU_j 's optimal PPL, and $r_{s,j}^0$ is SU_j 's expected transmit radius.

When there are two SUs applying for ch_i , we can decide SUs' MTR in the following three cases.

Case 1: Interference free circles do not intersect with each other, as Fig. 6 shows. In this case, both SU_1 and SU_2 can access the channel with its expected transmit radius.

Case 2: Interference free circles intersect with each other, as Fig. 7 shows. In this case, our goal is to maximize PU_i 's revenue

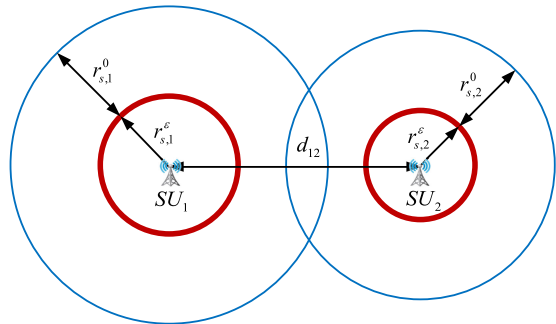


Fig. 7. Interference free circles intersect with each other.

$$T_{p,i} = \frac{R_1}{r_{s,1}^0} k_1 \pi (r_{s,1}^\epsilon + R_1)^2 + \frac{R_2}{r_{s,2}^0} k_1 \pi (r_{s,2}^\epsilon + R_2)^2. \quad (9)$$

Intuitively, we observe that PU_i can achieve the maximum utility if and only if the interference free circles of both SU_1 and SU_2 are tangent with each other. The reason is that if there is an allocation strategy which the interference free circles are not tangent with each other, we can always find a better allocation strategy by expanding an arbitrary SU's interference free circle to achieve better utility. Thus, we have

$$R_2 = d_{12} - r_{s,1}^\epsilon - r_{s,2}^\epsilon - R_1,$$

Define $M = d_{12} - r_{s,1}^\epsilon - r_{s,2}^\epsilon$, M is a constant. Thus, the best allocation strategy can be achieved by solving the following optimization problem:

Problem 6.1.

$$\begin{aligned} \arg \max_{R_1} T_{p,i} &= \frac{R_1}{r_{s,1}^0} k_1 \pi (r_{s,1}^\epsilon + R_1)^2 \\ &\quad + \frac{M - R_1}{r_{s,2}^0} k_1 \pi (r_{s,2}^\epsilon + M - R_1)^2, \\ \text{s.t.} \quad &0 \leq R_1 \leq r_{s,1}^0, \\ &M - r_{s,2}^0 \leq R_1 \leq M, \end{aligned}$$

Then, we show that Problem 6.1 has optimal solution and provide efficient method to achieve it.

Theorem 6.1. *There exists an optimal PPL for SU L_s^* to solve Problem 6.1.*

Proof. The second order derivative of R_p is

$$\frac{d^2 T_{p,i}}{dR_1^2} = \frac{4k_1 \pi}{r_{s,1}^0} (r_{s,1}^\epsilon + R_1) + \frac{2k_1 \pi}{r_{s,2}^0} (M - R_1).$$

Since $\frac{d^2 T_{p,i}}{dR_1^2}$ is always larger than zero, so that $T_{p,i}$ is a convex function. In addition, the constraint of Problem 6.1 is convex. Therefore, Problem 6.1 is a convex optimization problem, which have an optimal solution. \square

Since Problem 6.1 is a convex optimization problem, we can solve it by existing convex solver such as gradient decent method [37].

Case 3: Privacy preserving circles intersect with each other, as Fig. 8 shows. In this case, only one SU is allowed to access ch_i as the analysis in Section 4.3. The best allocation strategy is to choose the SU which brings more revenue to PU. After

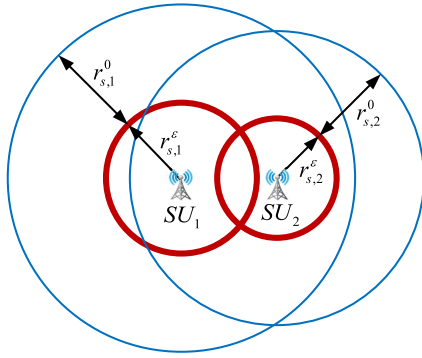


Fig. 8. Privacy preserving circles intersect with each other.

excluding one SU, the remaining one can transmit with its expected transmit radius. And PU_i 's maximum revenue is $T_{p,i}^* = \max\{T_{p,i}^1, T_{p,i}^2\}$.

6.2 The Channel is Occupied

When the channel is occupied, database should decide PU_i 's optimal PPL and each SU's MTR. Similar to Section 6.1, we only consider the two SUs situation.

Case 1: Interference free circles do not intersect with each other, as Fig. 9 shows.

To simplify the analysis, we consider a special situation where PU_i and two SUs are on a straight line as shown in Fig. 9. The other relative locations between PU_i and SUs can be transformed to the above situation, since PU_i 's decision is only relevant to the distance between PU_i and the two SUs.

When PU_i 's interference free circle is tangent with SU_j 's interference free circle, we have

$$\begin{aligned} T_{p,i}^1 &= \frac{R_1}{r_{s,1}^0} k_1 \pi (r_{s,1}^\epsilon + R_1)^2 \\ &= \frac{k_1 \pi}{r_{s,1}^0} (d_{m1} - r_{s,1}^\epsilon - L_p)(d_{m1} - L_p)^2, \\ T_{p,i}^2 &= \frac{R_2}{r_{s,2}^0} k_1 \pi (r_{s,2}^\epsilon + R_2)^2, \\ &= \frac{k_1 \pi}{r_{s,2}^0} (d_{m2} - r_{s,2}^\epsilon - L_p)(d_{m2} - L_p)^2. \end{aligned}$$

In this case, PU_i 's utility function is piecewise, and is relevant to the relative locations of point A , B , C and D . The reason is that when L_p extends from 0, the interference free circle of PU_i will intersect with the two SUs' circles in point A , B , C and D in different order. And for each order, PU_i 's utility function is different. We will discuss all the possible orders as follows:

(a) The order is $B \rightarrow C \rightarrow A \rightarrow D$.

$$E[U_p] = \begin{cases} T_{p,i}^1 + k_1 \pi (r_{s,2}^\epsilon + r_{s,2}^0)^2 - \frac{k_p}{L_p} & \text{if } d_{m1} - r_{s,1}^\epsilon - r_{s,1}^0 \leq L_p \leq d_{m2} - r_{s,2}^\epsilon - r_{s,2}^0 \\ T_{p,i}^1 + T_{p,i}^2 - \frac{k_p}{L_p} & \text{if } d_{m2} - r_{s,2}^\epsilon - r_{s,2}^0 < L_p \leq d_{m1} - r_{s,1}^\epsilon \\ T_{p,i}^2 - \frac{k_p}{L_p} & \text{if } d_{m1} - r_{s,1}^\epsilon < L_p \leq d_{m2} - r_{s,2}^\epsilon \end{cases}$$

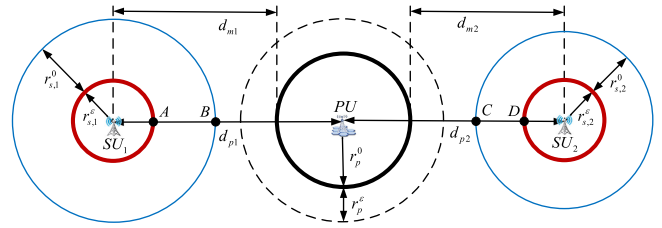


Fig. 9. Interference free circles do not intersect with each other.

(b) The order is $B \rightarrow C \rightarrow D \rightarrow A$.

$$E[U_p] = \begin{cases} T_{p,i}^1 + k_1 \pi (r_{s,2}^\epsilon + r_{s,2}^0)^2 - \frac{k_p}{L_p} & \text{if } d_{m1} - r_{s,1}^\epsilon - r_{s,1}^0 \leq L_p \leq d_{m2} - r_{s,2}^\epsilon - r_{s,2}^0 \\ T_{p,i}^1 + T_{p,i}^2 - \frac{k_p}{L_p} & \text{if } d_{m2} - r_{s,2}^\epsilon - r_{s,2}^0 < L_p \leq d_{m2} - r_{s,2}^\epsilon \\ T_{p,i}^1 - \frac{k_p}{L_p} & \text{if } d_{m2} - r_{s,2}^\epsilon < L_p \leq d_{m1} - r_{s,1}^\epsilon \end{cases}$$

(c) The order is $B \rightarrow A \rightarrow C \rightarrow D$.

$$E[U_p] = \begin{cases} T_{p,i}^1 + k_1 \pi (r_{s,2}^\epsilon + r_{s,2}^0)^2 - \frac{k_p}{L_p} & \text{if } d_{m1} - r_{s,1}^\epsilon - r_{s,1}^0 \leq L_p \leq d_{m1} - r_{s,1}^\epsilon \\ k_1 \pi (r_{s,2}^\epsilon + r_{s,2}^0)^2 - \frac{k_p}{L_p} & \text{if } d_{m1} - r_{s,1}^\epsilon < L_p \leq d_{m2} - r_{s,2}^\epsilon - r_{s,2}^0 \\ T_{p,i}^2 - \frac{k_p}{L_p} & \text{if } d_{m2} - r_{s,2}^\epsilon - r_{s,2}^0 < L_p \leq d_{m2} - r_{s,2}^\epsilon \end{cases}$$

Since other orders, i.e., PU_i 's interference free circle first intersect with C rather than B , is symmetric with the above three orders, we will not list here.

Our goal is to decide the optimal L_p^* to maximize $E[U_p]$. To solve this optimization problem, we can find the optimal point in each interval of the objective function, and choose the point which achieves the global optimum. We observe that the objective function is similar to Problem 5.4 in each interval. Thus, we can find the optimal point in each interval as the analysis in Problem 5.4.

Case 2: Interference free circles intersect with each other, as Fig. 10 shows.

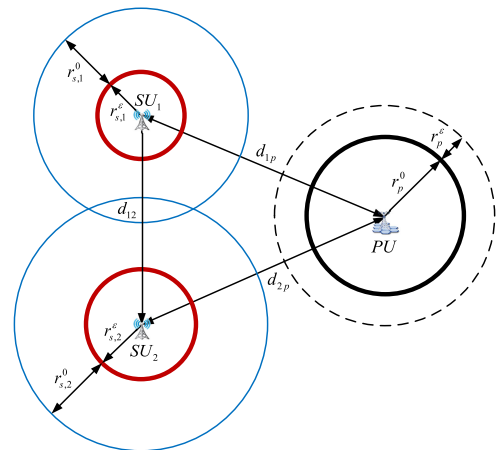


Fig. 10. Interference free circles intersect with each other.

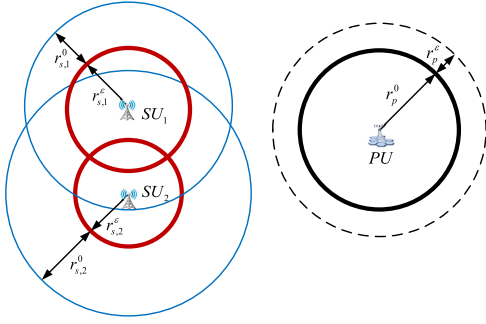


Fig. 11. Privacy preserving circles intersect with each other.

In this case, we need to optimize PU_i 's PPL and SUs' MTR. At the same time, we need to guarantee that the interference free circles of both PU_i and SUs do not intersect with each other. To optimize PU_i 's utility, there are two situations: One is that only one SU is available, i.e., the other SU's MTR is 0. The other is that both two SUs are available.

When only one SU is available, we need to find PU_i 's optimal PPL in the situation where only SU_1 or SU_2 exist (similar to the analysis in Section 5), and we denote the optimal PPL as $L_{p,1}^*$ and $L_{p,2}^*$, respectively.

When both two SUs are available, we need to solve the following optimization problem

Problem 6.2.

$$\begin{aligned} \arg \max_{R_1, R_2, L_p} E[U_p] &= k_1 \pi \left[\frac{R_1}{r_{s,1}^0} (r_{s,1}^\epsilon + R_1)^2 + \frac{R_2}{r_{s,2}^0} (r_{s,2}^\epsilon + R_2)^2 \right] \\ &\quad - \frac{k_p}{L_p} \\ \text{s.t.} \quad &R_1 + r_{s,1}^\epsilon + r_p + L_p \leq d_{1p} \\ &R_2 + r_{s,2}^\epsilon + r_p + L_p \leq d_{2p} \\ &R_1 + r_{s,1}^\epsilon + R_2 + r_{s,2}^\epsilon \leq d_{12} \\ &0 \leq R_1 \leq r_{s,1}^0 \\ &0 \leq R_2 \leq r_{s,2}^0 \\ &L_p > 0 \end{aligned}$$

Since the objective function is nonconvex, we will adopt the gradient decent method [37] to find an approximate solution. After finding the optimal PPL L_p^* , we can decide the global optimal PPL

$$L_p^* = \max\{L_{p,1}^*, L_{p,2}^*, L_p^*\}.$$

Case 3: Privacy preserving circles intersect with each other as Fig. 11 shows.

In this case, similar to the analysis in Case 3 of channel vacant situation, we only need to satisfy one SU's requirement and exclude the other one to avoid interference. To achieve the maximum utility, PU can exclude one SU and decide its optimal PPL separately, and then choose the SU which can achieve the maximum utility. If only one SU exist, we decide PU_i 's optimal PPL $L_{p,1}^*$ and $L_{p,2}^*$, respectively. And PU_i 's global optimal PPL is given by

$$L_p^* = \max\{L_{p,1}^*, L_{p,2}^*\}.$$

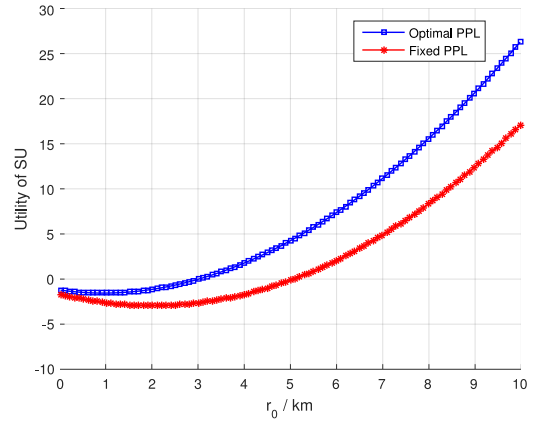


Fig. 12. Optimal PPL versus fixed PPL to SU's utility.

7 SIMULATION

In this section, we first introduce the simulation setup. Then, we illustrate the advantage of our proposed UMax protocol by comparing with the baseline protocol.

7.1 Simulation Setup

First, we present the baseline protocol: different from the proposed UMax protocol where SUs and PUs decide their optimal PPLs before querying the database and deciding the allocation strategy, both PUs and SUs in the baseline protocol choose a predefined fixed PPLs.

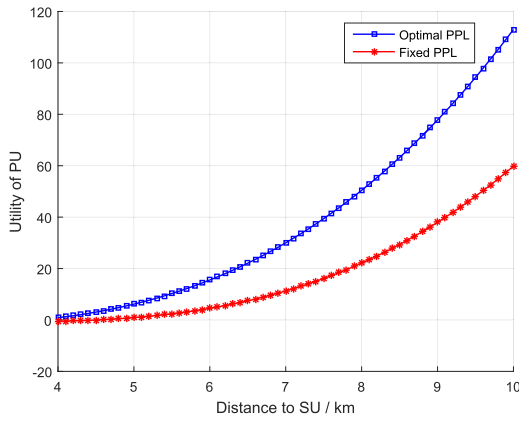
For brevity, we set the unit of the distance in the simulation to *kilometer*, and only utilize the number to describe the distance in the following part. We set the fixed PPLs of SUs and PUs in the baseline protocol to be 2 and 1, respectively. The privacy cost coefficient k_s and k_p are set to be 1 and 2, k_1 and k_2 are set to be 0.1 and 0.2.

Then, we introduce the simulation settings: 1) We compare SUs' utilities when they choose different required transmit radius; 2) PUs' utilities are compared in different location settings, i.e., the relative distance between PUs and SUs are different. 3) We construct an $20 * 20$ area and randomly deploy 10 PUs and 20 SUs, we then compare the utilities of different PUs.

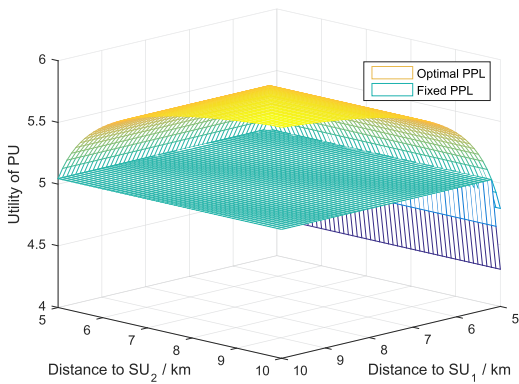
7.2 Performance Comparison

Fig. 12 compares the utilities of the proposed protocol with the baseline protocol for SUs. The simulation result shows that the proposed protocol can improve SUs' utility efficiently for different required transmit radius r_0 . Notice that SU's utility is negative when r_0 is shorter than a threshold, e.g., 3 for optimal PPL and 5 for fixed PPL. The reason is that when r_0 is shorter than a threshold, SU's revenue R_s is few which is obvious in Problem 5.3. In another hand, SUs' should choose a small PPL to guarantee their privacy. As a result, SU's payment C_{buy} would be larger than its revenue R_s which lead to a negative utility.

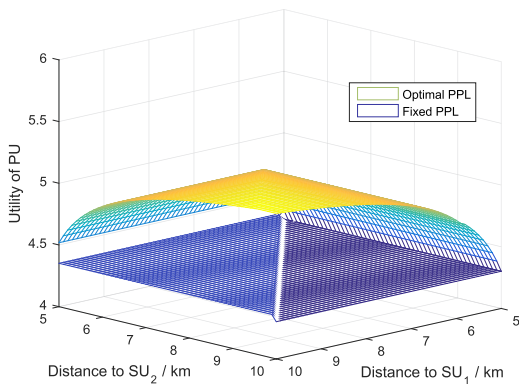
Fig. 13 compares the utilities of the proposed protocol with the baseline protocol for PUs in different location settings. The simulation is conducted in three different scenarios: (i) Single PU single SU scenario; (ii) Single PU two SUs scenario where SUs do not intersect with each other; (iii) Single PU two SUs scenario where SUs intersect with each other.



(a) Single PU single SU scenario.



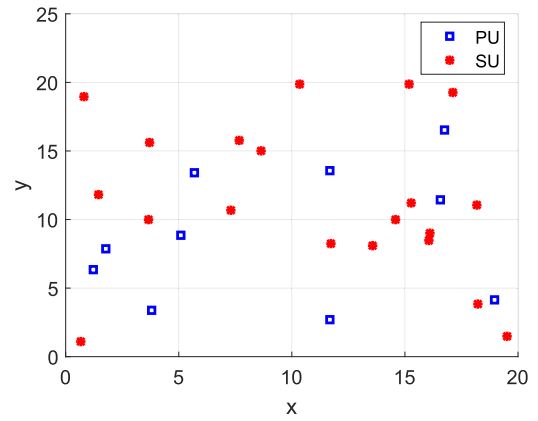
(b) Single PU two SUs scenario where SUs do not intersect with each other.



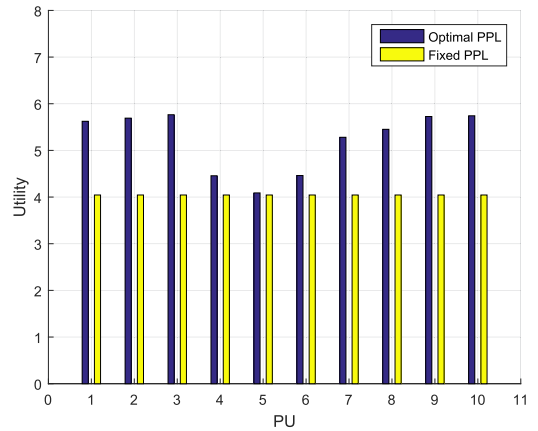
(c) Single PU two SUs scenario where SUs intersect with each other.

Fig. 13. Optimal PPL versus fixed PPL to PU's utility.

Fig. 13a shows that, in the single PU single SU scenario, PUs in the proposed protocol achieves higher utility compared with the baseline where PUs choose the fixed PPL. In another hand, PUs achieves higher utility when the distance between PU and SU increase. The reason is that, PUs would receive more revenue through selling spectrum to SUs when the distance increase. Figs. 13b and 13c illustrate the advantage of the proposed protocol where there are one PU and two SUs, where x coordinate refers to the distance between PU and SU_1 , and y coordinate refers to the distance between PU and SU_2 . Specifically, Fig. 13b shows the scenario where SUs do not intersect with each other, the theoretic analysis of which is given in Section 6.2



(a) Distribution map of PUs and SUs



(b) Utility of different PUs.

Fig. 14. PUs' utilities with random deployment.

Case 1, and Fig. 13c shows the scenario where SUs intersect with each other, the theoretic analysis of which is given in Section 6.2 Case 2. Both Figs. 13b and 13c shows that PUs achieves higher utility compared with the baseline where PUs choose fixed PPLs.

Fig. 14 compares PUs' utilities of the proposed protocol with the baseline protocol when we randomly deploy some PUs and SUs in a given area. In Fig. 14a, we randomly deploy 10 PUs and 20 SUs in an 20×20 area. Fig. 14b compares the utilities of different PUs. The simulation result shows that the proposed protocol achieves higher utility compared with the baseline protocol for all PUs.

8 CONCLUSION

In this paper, we proposed a novel location privacy preservation scheme, while achieving bilateral utilization maximization of both PUs and SUs. First, a quantitative mechanism was proposed to preserve the location privacy of both PUs and SUs simultaneously based on the concept of differential privacy. Based on the quantitative mechanism framework, we further proposed a novel privacy preserving Utility Maximization protocol (UMax). UMax allows for both PUs and SUs to adjust their privacy preserving levels to achieve the optimal utility. Extensive simulations demonstrated that our proposed mechanism can efficiently increase both PUs' and SUs' utility while preserving their location privacy.

ACKNOWLEDGMENTS

This work is partially supported by NSFC under Grant 61731004, by ZJU-SUTD joint project under Grant 188170-11102/012, by Nature Science Foundation of Jiangsu Province under Grant BK20171264.

REFERENCES

- [1] Y.-C. Liang, K.-C. Chen, G. Y. Li, and P. Mahonen, "Cognitive radio networking and communications: An overview," *IEEE Trans. Veh. Technol.*, vol. 60, no. 7, pp. 3386–3407, Sep. 2011.
- [2] R. Deng, J. Chen, X. Cao, Y. Zhang, S. Maharjan, and S. Gjessing, "Sensing-performance tradeoff in cognitive radio enabled smart grid," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 302–310, Mar. 2013.
- [3] J. Chen, Q. Yu, B. Chai, Y. Sun, Y. Fan, and X. Shen, "Dynamic channel assignment for wireless sensor networks: A regret matching based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 1, pp. 95–106, Jan. 2015.
- [4] J. Wang, M. Ghosh, and K. Challapali, "Emerging cognitive radio applications: A survey," *IEEE Commun. Mag.*, vol. 49, no. 3, pp. 74–81, Mar. 2011.
- [5] M. Patel and J. Wang, "Applications, challenges, and prospective in emerging body area networking technologies," *IEEE Wireless Commun.*, vol. 17, no. 1, pp. 80–88, Feb. 2010.
- [6] Ex-parte comments of ge healthcare in docket 06–135. [Online]. Available: <http://fjallfoss.fcc.gov/ecfs/document/view?id=6519820996/>, Dec. 2007.
- [7] X. Duan, C. Zhao, S. He, P. Cheng, and J. Zhang, "Distributed algorithms to compute walrasian equilibrium in mobile crowdsensing," *IEEE Trans. Ind. Electronics*, vol. 64, no. 5, pp. 4048–4057, May 2017.
- [8] A. S. Rawat, P. Anand, H. Chen, and P. K. Varshney, "Collaborative spectrum sensing in the presence of byzantine attacks in cognitive radio networks," *IEEE Trans. Signal Process.*, vol. 59, no. 2, pp. 774–786, Feb. 2011.
- [9] J. Chen, J. Li, S. He, T. He, Y. Gu, and Y. Sun, "On energy-efficient trap coverage in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 10, no. 1, 2013, Art. no. 2.
- [10] J. Chen, J. Li, and T. H. Lai, "Energy-efficient intrusion detection with a barrier of probabilistic sensors: Global and local," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4742–4755, Sep. 2013.
- [11] J. Chen, J. Li, and T. H. Lai, "Trapping mobile targets in wireless sensor networks: An energy-efficient perspective," *IEEE Trans. Veh. Technol.*, vol. 62, no. 7, pp. 3287–3300, Sep. 2013.
- [12] G. Yang, S. He, Z. Shi, and J. Chen, "Promoting cooperation by the social incentive mechanism in mobile crowdsensing," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 86–92, Mar. 2017.
- [13] Y. Zhao, J. Gaedder, K. K. Bae, and J. H. Reed, "Radio environment map enabled situation-aware cognitive radio learning algorithms," in *Proc. SDR Forum Tech. Conf.*, 2006, pp. 1–6.
- [14] S. N. Khan, M. A. Kalil, and A. Mitschele-Thiel, "Distributed resource map: A database-driven network support architecture for cognitive radio ad hoc networks," in *Proc. 4th Int. Congress Ultra Modern Telecommun. Control Syst. Workshops*, 2012, pp. 188–194.
- [15] R. Murty, R. Chandra, T. Moscibroda, and P. Bahl, "Senseless: A database-driven white spaces network," *IEEE Trans. Mobile Comput.*, vol. 11, no. 2, pp. 189–203, Feb. 2012.
- [16] Z. Gao, H. Zhu, Y. Liu, M. Li, and Z. Cao, "Location privacy in database-driven cognitive radio networks: Attacks and countermeasures," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2013, pp. 2751–2759.
- [17] B. Bahrak, S. Bhattarai, A. Ullah, J.-M. J. Park, J. Reed, and D. Gurney, "Protecting the primary users' operational privacy in spectrum sharing," in *Proc. IEEE Int. Symp. Dynamic Spectrum Access Netw.*, 2014, pp. 236–247.
- [18] C. Dwork, "Differential privacy," in *Proc. 33rd Int. Conf. Automata Languages Program.*, 2006, pp. 1–12.
- [19] R. Chen, J.-M. Park, and K. Bian, "Robust distributed spectrum sensing in cognitive radio networks," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2008, pp. 1876–1884.
- [20] A. W. Min, X. Zhang, and K. G. Shin, "Detection of small-scale primary users in cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 2, pp. 349–361, Feb. 2011.
- [21] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, and S. He, "Narrow-band internet of things: Implementations and applications," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2309–2314, Dec. 2017.
- [22] S. He, J. Chen, F. Jiang, D. K. Yau, G. Xing, and Y. Sun, "Energy provisioning in wireless rechargeable sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 10, pp. 1931–1942, Oct. 2013.
- [23] R. Chen, J.-M. Park, and J. H. Reed, "Defense against primary user emulation attacks in cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, pp. 25–37, Jan. 2008.
- [24] A. W. Min, K. G. Shin, and X. Hu, "Secure cooperative sensing in IEEE 802.22 WRANS using shadow fading correlation," *IEEE Trans. Mobile Comput.*, vol. 10, no. 10, pp. 1434–1447, Oct. 2011.
- [25] P. Shankar, V. Ganapathy, and L. Iftode, "Privately querying location-based services with sybilquery," in *Proc. ACM 12th ACM Int. Conf. Ubiquitous Comput.*, 2009, pp. 31–40.
- [26] R. Shokri, C. Troncoso, C. Diaz, J. Freudiger, and J.-P. Hubaux, "Unraveling an old cloak: K-anonymity for location privacy," in *Proc. 7th ACM Workshop Privacy Electron. Soc.*, 2010, pp. 115–118.
- [27] A. Khoshgozaran, C. Shahabi, and H. Shirani-Mehr, "Location privacy: Going beyond k-anonymity, cloaking and anonymizers," *Knowl. Inform. Syst.*, vol. 26, no. 3, pp. 435–465, 2011.
- [28] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkatasubramanian, "l-diversity: Privacy beyond k-anonymity," *ACM Trans. Knowl. Discovery Data*, vol. 1, no. 1, 2007, Art. no. 3.
- [29] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. IEEE 25th Int. Conf. Data Eng.*, 2007, pp. 106–115.
- [30] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *Advances in Spatial and Temporal Databases*. Berlin, Germany: Springer, 2007, pp. 239–257.
- [31] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: Anonymizers are not necessary," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2008, pp. 121–132.
- [32] D. He, N. Kumar, S. Zeadally, A. Vinel, and L. T. Yang, "Efficient and privacy-preserving data aggregation scheme for smart grid against internal adversaries," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2411–2419, Sep. 2017.
- [33] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proc. 23rd ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 901–914.
- [34] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Theoretical Comput. Sci.*, vol. 9, no. 3–4, pp. 211–407, 2013.
- [35] H. Zhang, Y. Shu, P. Cheng, and J. Chen, "Privacy and performance trade-off in cyber-physical systems," *IEEE Netw.*, vol. 30, no. 2, pp. 62–66, Mar./Apr. 2016.
- [36] A. Ghosh and A. Roth, "Selling privacy at auction," *Games and Econ. Behavior*, vol. 91, pp. 334–346, 2015.
- [37] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.
- [38] R. Garver, "On the nature of the roots of a quartic equation," *Mathematics News Letter*, vol. 3, no. 4, pp. 6–8, Jan. 1933.

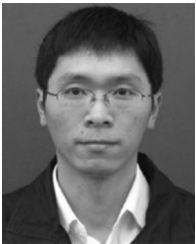


Zhikun Zhang received the BEng degree in automation from Shandong University, Jinan, China, in 2014. From Oct. 2017 to Oct. 2018, he is a visiting scholar with Purdue University, West Lafayette, IN. He is currently working toward the PhD degree in the Group of Networked Sensing and Control (IIPC-NeSC) in the State Key Laboratory of Industrial Control Technology, Zhejiang University. His research interests include location privacy, differential privacy and its applications in cognitive radio, crowdsensing system, and machine learning. He is a student member of the IEEE.



Heng Zhang received the PhD degree in control science and engineering from Zhejiang University, in 2015. He is currently an associate professor in the School of Science, Huaihai Institute of Technology, Lianyungang, Jiangsu, China. He is also a research fellow with Western Sydney University. His research interests include security and privacy in cyber-physical systems, control and optimization theory. He is an editor board member of several academic journals, including the *IET Wireless Sensor Systems*, the *EURASIP Journal on Wireless Communications and Networking*, the *KSII Transactions on Internet and Information Systems*, etc. He also serves as a guest editor of the *Journal of The Franklin Institute*, the *Peer-to-Peer Networking and Applications*. He is also an active reviewer of the *IEEE Transactions on Automatic Control*, the *IEEE Transactions on Control of Network Systems*, the *IEEE Transactions on Information Forensics and Security*, and the *IEEE Transactions on Wireless Communications*, etc. He is a member of the IEEE.

Journal on Wireless Communications and Networking, the *KSII Transactions on Internet and Information Systems*, etc. He also serves as a guest editor of the *Journal of The Franklin Institute*, the *Peer-to-Peer Networking and Applications*. He is also an active reviewer of the *IEEE Transactions on Automatic Control*, the *IEEE Transactions on Control of Network Systems*, the *IEEE Transactions on Information Forensics and Security*, and the *IEEE Transactions on Wireless Communications*, etc. He is a member of the IEEE.



Shibo He (M'13) received the PhD degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2012. From Nov. 2010 to Nov. 2011, he was a visiting scholar with the University of Waterloo, Waterloo, ON, Canada. He was an associate research scientist from March 2014 to May 2014, and a postdoctoral scholar from May 2012 to February 2014, with Arizona State University, Tempe, AZ. He is currently a professor with Zhejiang University. His research interests include wireless sensor networks,

crowdsensing and big data analysis. He serves on the editorial board of the *IEEE Transactions on Vehicular Technology*, the *Springer Peer-to-Peer Networking and Application*, the *KSII Transactions Internet and Information Systems*, and is a guest editor of the *Elsevier Computer Communications* and the *Hindawi International Journal of Distributed Sensor Networks*. He served as publicity chair of the IEEE SECON 2016, Registration and Finance chair of the ACM MobiHoc 2015, TPC co-chair of the IEEE ScalCom 2014, TPC vice co-chair of the ANT 2013-2014, Track co-chair of the Pervasive Algorithms, Protocols, and Networks of EUSPN 2013, Web co-chair of the IEEE MASS 2013, and Publicity co-chair of the IEEE WiSARN 2010. He is the recipient of IEEE Asia-Pacific outstanding researcher award, 2015. He is a member of the IEEE.






Peng Cheng (M'10) received the BE degree in automation, and the PhD degree in control science and engineering, both from Zhejiang University, Hangzhou, P. R. China, in 2004 and 2009 respectively. Currently he is a professor in the Department of Control Science and Engineering, Zhejiang University. He serves as associate editor of the *Wireless Networks*, the *International Journal of Distributed Sensor Networks*, and the *International Journal of Communication systems*. He also served as publicity co-chair of the IEEE

MASS 2013 and Local Arrangement chair of the ACM MobiHoc 2015. His research interests include networked sensing and control, cyber-physical systems, and robust control. He is a member of the IEEE.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Cross-Rack-Aware Single Failure Recovery for Clustered File Systems

zhirong shen , Patrick P. C. Lee , Jiwu Shu, *Senior Member, IEEE*, and Wenzhong Guo 

Abstract—How to improve the performance of single failure recovery has been an active research topic because of its prevalence in large-scale storage systems. We argue that when erasure coding is deployed in a clustered file system (CFS), existing single failure recovery designs are limited in different aspects: neglecting the bandwidth diversity property in a CFS architecture, targeting specific erasure code constructions, and no special treatment on load balancing during recovery. In this paper, we propose CAR, a *cross-rack-aware recovery* algorithm that is designed to improve the performance of single failure recovery of a CFS that employs Reed-Solomon codes for general fault tolerance. For each stripe, CAR finds a recovery solution that retrieves data from the minimum number of racks. It also reduces the amount of cross-rack repair traffic by performing intra-rack data aggregation prior to cross-rack transmission. Furthermore, by considering multi-stripe recovery, CAR balances the amount of cross-rack repair traffic across multiple racks. Evaluation results show that CAR can effectively reduce the amount of cross-rack repair traffic and the resulting recovery time.

Index Terms—Cross-rack bandwidth, single failure recovery, erasure coding, load balancing, partial decoding, clustered file systems

1 INTRODUCTION

TO process an ever-increasing amount of data, distributed computing applications often build on a *clustered file system (CFS)*, which provides a unified and scalable storage platform. A CFS is constructed over a number of physically independent storage servers, which we refer to as *nodes* in this paper. Examples of CFSes include Google File System [13], Hadoop Distributed File System [37], and Windows Azure Storage [5].

Failures are commonplace in large-scale CFSes [8], [11], [13], [32], [33]. In particular, most failures found in CFSes are *single node failures* (or single failures in short), which can occupy over 90 percent of all failure events in real deployment [11]. To maintain data availability, a common approach is to store data with redundancy. Compared with traditional replication, *erasure coding* is shown to achieve higher fault tolerance with less redundancy [38], and hence is increasingly used in today's CFSes for improved storage efficiency. The mainstream approach of erasure coding takes original pieces of information of the same size (termed *data chunks*) as inputs and produces redundant pieces of information that are also of the same size (termed *parity chunks*), such that if a data or parity chunk is lost, we can still retrieve other available data and parity chunks to

reconstruct the lost chunk. The collection of data and parity chunks that are connected by the erasure coding forms a *stripe*. A CFS stores multiple stripes of information, each of which is independently encoded.

Given the prevalence of single failures, there have been a spate of solutions (e.g., [12], [18], [28], [29], [35], [40], [42], [43]) on improving the performance of single failure recovery in erasure-coded storage systems. The main idea of such solutions is to selectively retrieve different portions of data and parity chunks within a stripe, with a common objective of minimizing the amount of *repair traffic* (i.e., the amount of information retrieved from surviving nodes for data reconstruction).

On the other hand, when we examine existing single failure recovery solutions, there remain three limitations (see Section 2.4 for details). First, typical CFS architectures organize nodes in multiple racks, yet existing studies on single failure recovery neglect the *bandwidth diversity* property between intra-rack and cross-rack connections in a CFS architecture. Second, many single failure recovery solutions (e.g., [12], [40], [42], [43]) focus on specific code constructions, but cannot be directly applied to today's CFSes (e.g., [1], [11], [24]) that employ Reed-Solomon (RS) codes [30] for general fault tolerance. Third, existing single failure recovery solutions do not consider the load balancing issue during the recovery operation.

To address the above limitations, we reconsider the single failure recovery problem in a CFS setting. First, we should specifically minimize the *cross-rack repair traffic* (i.e., the amount of data to be retrieved across racks for data reconstruction), which plays an important role in improving the performance of single failure recovery with regard to the scarce cross-rack bandwidth in a CFS. Second, our single failure recovery design should address general fault tolerance (e.g., based on RS codes). Finally, we should balance the cross-rack repair

- Z. Shen and P.P.C. Lee are with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. E-mail: zhirong.shen2601@gmail.com, pcleee@cse.cuhk.edu.hk.
- J. Shu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: shujw@tsinghua.edu.cn.
- W. Guo is with the College of Mathematics and Computer Science, Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou Shi, Fujian Sheng 350002, China. E-mail: guowenzhong@fzu.edu.cn.

Manuscript received 17 Mar. 2017; revised 2 Nov. 2017; accepted 10 Nov. 2017. Date of publication 16 Nov. 2017; date of current version 18 Mar. 2020. (Corresponding author: Jiwu Shu.)
Digital Object Identifier no. 10.1109/TDSC.2017.2774299

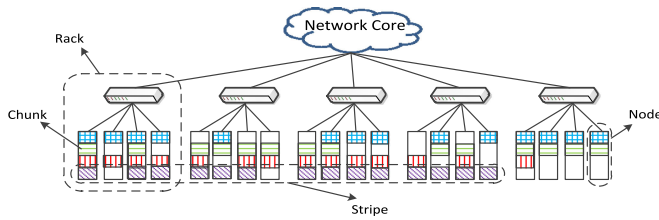


Fig. 1. Illustration of a CFS architecture, composed of five racks with four nodes each. The CFS contains four stripes of 14 chunks encoded by a $(k = 8, m = 6)$ code, in which the chunks with the same color and fill pattern belong to the same stripe. Note that the number of chunks in each node may be different.

traffic at the rack level (i.e., across multiple racks) while keeping the total amount of cross-rack repair traffic minimum, so as to ensure that the performance of single failure recovery is not bottlenecked by a single rack.

To this end, we propose *cross-rack-aware recovery (CAR)*, a new single failure recovery algorithm that aims to reduce and balance the amount of cross-rack repair traffic for a single failure recovery in a CFS that deploys RS codes for general fault tolerance. CAR has three key techniques. First, for each stripe, CAR examines the data layout and finds a recovery solution in which the resulting repair traffic comes from the minimum number of racks. Second, CAR performs intra-rack aggregation for the retrieved chunks in each rack before transmitting them across racks, so as to further reduce the amount of cross-rack repair traffic. Finally, CAR examines the per-stripe recovery solutions across multiple stripes, and constructs a multi-stripe recovery solution that balances the amount of cross-rack repair traffic across multiple racks.

Our contributions are summarized as follows.

- We reconsider the single failure recovery problem in the CFS setting, and identify the open issues that are not addressed by existing studies on single failure recovery.
- We propose CAR, a new cross-rack-aware single failure recovery algorithm for a CFS setting. CAR is designed based on RS codes. It reduces the amount of cross-rack repair traffic for each stripe, and additionally searches for a multi-stripe recovery solution that balances the cross-rack repair traffic across racks. We also discuss how CAR should be deployed to achieve general rack-level fault tolerance.
- We implement CAR and conduct extensive testbed experiments based on different CFS settings with up to 20 nodes. We show that CAR can reduce 66.9 percent of cross-rack repair traffic and 47.9 percent of recovery time when compared to a baseline single failure recovery design that does not consider the bandwidth diversity property of a CFS. Also, we show that CAR effectively balances the amount of cross-rack repair traffic across multiple racks.

The rest of this paper proceeds as follows. Section 2 presents the background details of erasure coding and reviews related work on single failure recovery. Section 3 formulates and motivates the problem in the CFS setting. Section 4 presents the design of CAR. Section 5 presents our evaluation results on CAR based on testbed experiments. Finally, Section 6 concludes the paper.

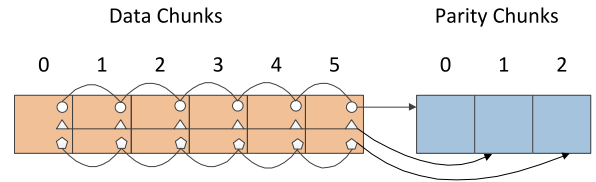


Fig. 2. Encoding of $(k = 6, m = 3)$ RS codes for a stripe, in which there are six data chunks and three parity chunks. If one of the data or parity chunks fails, any six surviving chunks within the stripe can be retrieved for reconstruction.

2 BACKGROUND AND RELATED WORK

2.1 Basics

This paper considers a special type of distributed storage system architecture called a *clustered file system*, which arranges storage nodes into *racks*, such that all nodes within the same rack are connected by a *top-of-rack (ToR) switch*, while all racks are connected by a *network core* that comprises layers of aggregation switches. Fig. 1 illustrates a CFS composed of five racks with four nodes each (i.e., 20 nodes in total). Some well-known distributed storage systems, such as Google File System [13], Hadoop Distributed File System [37], and Windows Azure Storage [5], realize the CFS architecture. Such a CFS architecture is also considered in the literature (e.g., [6], [20]).

We use erasure coding to maintain data availability for a CFS. We consider a popular family of erasure codes that are: (1) *Maximum Distance Separable (MDS)* codes, meaning that fault tolerance is achievable with the minimum storage redundancy (i.e., the optimal storage efficiency), and (2) *systematic*, meaning that the original data is retained after encoding. Specifically, we construct a (k, m) code (which is MDS and systematic) with two configurable parameters k and m . A (k, m) code takes k original (uncoded) data chunks of the same size as inputs and produces m (coded) parity chunks that are also of the same size, such that any k out of the $k + m$ chunks can sufficiently reconstruct all original data chunks. The $k + m$ chunks collectively form a *stripe*, and are distributed over $k + m$ different nodes. Note that the placement of chunks should also ensure rack-level fault tolerance [20], such that there are at least k chunks for data reconstruction in other surviving racks in the presence of rack failures.

For an erasure-coded CFS that stores a large amount of data, it contains multiple stripes of data/parity chunks that are independently encoded. In this case, each node stores a different number of chunks that belong to multiple stripes. For example, referring to the CFS in Fig. 1, there are four stripes spanning over 20 nodes, in which the leftmost node stores four chunks, while the rightmost node stores only two chunks.

2.2 Erasure Code Constructions

There have been various proposals on erasure code construction in the literature. Practical erasure codes often realize encoding/decoding operations based on the arithmetic over the Galois field [27]. Reed Solomon codes [30] are one representative example. RS codes are MDS, and support any pair of (k, m) in general. For example, Fig. 2 shows a stripe of the $(k = 6, m = 3)$ RS code, which contains six data chunks (i.e., $k = 6$) and three parity chunks (i.e., $m = 3$).

RS codes have been intensively used for erasure-coded storage in today's commercial storage systems for fault tolerance, such as Google's ColossusFS [1] and Facebook's HDFS [4]. In this paper, we design our CAR based on RS codes.

XOR-based erasure codes are a special family of erasure codes that perform encoding/decoding with XOR operations only. Examples of XOR-based erasure codes include RDP Code [7], X-Code [41], STAR Code [17], and HV Code [34]. XOR-based erasure codes are generally MDS, but they often have specific restrictions on the parameters k and m . For example, RDP Code [7] requires $(k = p - 1, m = 2)$, X-Code [41] requires $(k = p - 2, m = 2)$, and STAR Code [17] requires $(k = p, m = 3)$, where p is a prime number. Thus, XOR-based erasure codes are mainly used in local disk arrays.

Single failures (e.g., a single node failure or a single lost chunk within a stripe) are known to be the most common failure events in a CFS [11], [16]. In RS codes, k chunks are needed to be retrieved to recover a single lost chunk. Some erasure codes are specially designed for improving the performance of recovering a single failure. Regenerating codes [10] minimize the amount of repair traffic by allowing other surviving nodes to send computed data for data reconstruction, and achieve the optimal tradeoff between the level of storage redundancy and the amount of repair traffic. In particular, minimum-storage regenerating (MSR) codes [10] are MDS, and they minimize the amount of repair traffic subject to the minimum storage redundancy. Rashmi et al. [28] propose a new MSR code construction that also minimizes the amount of I/Os. Huang et al. [16] and Sathiamoorthy et al. [31] develop local reconstruction codes to reduce the amount of repair traffic, while incurring slightly more storage redundancy (and hence the codes are non-MDS).

Recent erasure codes address mixed failures (e.g., a combination of disk failures and sector errors) in a storage efficient way. Examples are SD codes [25] and STAIR Codes [19]. They are non-MDS, and perform encoding/decoding operations over the Galois field. They are mainly designed for local disk arrays.

2.3 Single Failure Recovery

There have been extensive studies in the literature that focus on improving the performance of single failure recovery. In addition to new erasure code constructions such as regenerating codes and local reconstruction codes (see Section 2.2), previous studies (e.g., [12], [18], [22], [40], [43]) pay close attention to XOR-based erasure codes. To reconstruct a lost chunk, the core idea of their proposals is to examine the relationship between the data and parity chunks of a stripe and then read different portions of a stripe, so as to minimize the amount of I/Os to access the storage nodes, and hence the amount of repair traffic, in single failure recovery. Some previous studies target specific XOR-based erasure code constructions. For example, Xiang et al. [40] and Xu et al. [42] prove the theoretical lower bound on the amount of I/Os for a single failure recovery for RDP Code and X-Code, respectively, both of which tolerate two node failures.

Some previous studies focus on minimizing the amount of I/Os for single failure recovery for general XOR-based erasure codes. Khan et al. [18] propose to enumerate all possible single failure recovery solutions and select the one that minimizes the amount of I/Os. Luo et al. [22] and Fu et al.

[12] extend the enumeration approach of Khan et al. [18] to balance the amount of I/Os to be read from surviving disks. Note that the enumeration approach is generally NP-hard. Thus, Zhu et al. [43] and Shen et al. [35] propose a greedy algorithm to search for the single failure recovery solution with the near-minimum amount of I/Os, while still supporting general XOR-based erasure codes.

Some studies also address the performance issue when deploying erasure codes in a CFS. For example, Li et al. [20] propose an efficient replica placement algorithm in a CFS to reduce the amount of cross-rack traffic when transforming replicated data to erasure-coded data. Xia et al. [39] present a new approach to switch between two erasure codes to balance the storage overhead and recovery performance.

There are recent studies that are closely related to ours. Mitra et al. [23] propose *Partial Parallel Repair (PPR)*, which decomposes a recovery operation into many small partial operations and schedules those partial operations in parallel, so as to achieve faster data recovery. The use of partial recovery operations is similar to our partial decoding approach (see Section 4.2), but PPR does not address how to minimize the cross-rack repair traffic. Li et al. [21] propose to minimize the single failure recovery time by dividing chunks into small slices and pipelining the repair of these slices. Hu et al. [14], [15] design Double Regenerating Codes (DRC) to use two stages to minimize the cross-rack repair traffic, while achieving the minimum storage efficiency. Their objective is similar to ours: Hu et al. [14], [15] focus on deriving a new erasure code construction that matches the optimal point of regenerating codes [10], while our work specifically focuses on RS codes, which have been widely deployed in current CFSes, and accordingly proposes inherently different recovery techniques.

2.4 Open Issues

In summary, there have been extensive studies on improving the performance of single failure recovery when deploying erasure coding in disk arrays or CFSes. On the other hand, we identify three open issues that are still unexplored when reconsidering the single failure recovery problem in a CFS setting. We have provided an overview of the open issues in Section 1, and following discussion provides more detailed explanations.

2.4.1 Lack of Considerations on Cross-Rack Repair Traffic

Existing single failure recovery optimizations [12], [18], [22], [35], [40], [42], [44], while significantly reducing the amount repair traffic, do not differentiate intra-rack and cross-rack data transmissions during recovery. In particular, a CFS architecture exhibits the property of *bandwidth diversity*, in which intra-rack bandwidth is considered to be sufficient, while cross-rack bandwidth is often *over-subscribed*. The typical values of the over-subscription ratio in a data center network are in the range from 5 to 20 [2]. Thus, cross-rack bandwidth is often considered to be a scarce resource [6], [9], [20]. A recovery solution that triggers a large amount of cross-rack traffic will unavoidably delay data reconstruction. How to minimize the amount of cross-rack repair traffic (i.e., the amount of data traffic triggered during recovery) should be carefully studied in a CFS setting.

2.4.2 Ineffectiveness for RS Codes

Most existing studies [12], [18], [22], [35], [40], [42], [44] on single failure recovery mainly focus on XOR-based erasure codes. While XOR-based erasure codes achieve high encoding/decoding performance by only using XOR operations, they are not the common choice in a CFS due to their specific fault tolerance settings (e.g., RDP codes [7] are RAID-6 codes that are double-fault-tolerant). In view of generality and flexibility, today's CFSes (e.g., [1], [11], [24]) usually employ Reed-Solomon codes [30] for general fault tolerance. RS codes perform encoding/decoding operations over finite fields [27], and have inherently different constructions from XOR-based erasure codes. In general, RS codes reconstruct a lost chunk by retrieving any k surviving chunks within the same stripe. This strategy implies that there are a maximum of $C(k+m-1, k)$ possible single failure recovery solutions (i.e., the number of combinations of selecting k out of $k+m-1$ surviving chunks). Furthermore, how to select the one with the minimum cross-rack repair traffic remains unexplored.

2.4.3 Load Balancing of Cross-Rack Repair Traffic in a Multi-Stripe Setting

Recall that a CFS often organizes data in multiple stripes, each of which is independently encoded (see Section 2.1). Existing single failure recovery solutions mainly focus on a single stripe. It is possible to further improve load balancing of a single failure recovery if we can consider a multi-stripe setting [12], [35], [42]. However, the load balancing schemes [12], [35], [42] only target XOR-based erasure codes, and also do not address the bandwidth diversity issue in a CFS. In a CFS, we are interested in balancing the amount of cross-rack repair traffic across multiple racks. However, solving single failure recovery problem for RS codes in a multi-stripe setting is non-trivial. As discussed above, a single failure recovery solution for a single stripe has a maximum of $C(k+m-1, k)$ possible options. If we consider $s > 1$ stripes, then the total number of possible options will increase to $[C(k+m-1, k)]^s$. How to efficiently search for a multi-stripe single failure recovery solution will be critical.

3 PROBLEM FORMULATION

This paper aims to address the following problem: *Given a CFS that deploys RS codes, can we simultaneously minimize and balance the amount of cross-rack repair traffic when we perform single failure recovery in the CFS?* In this section, we formulate the single failure recovery problem in a CFS setting. Table 1 summarizes the major notation used in this paper.

Consider a CFS that deploys a (k, m) RS code over r racks denoted by $\{A_1, A_2, \dots, A_r\}$. Suppose that a node fails, and we need to reconstruct the lost chunks in the failed node. Each stripe contains exactly one lost chunk. To make our analysis general, we assume that the lost chunks to be reconstructed in the failed node span $s \geq 1$ stripes. We denote the rack that contains the failed node as A_f ($1 \leq f \leq r$); also, we call the remaining racks (aside A_f) to be *intact racks* since the data stored in all their nodes remains intact. To repair the lost chunks in the failed node, suppose the cross-rack repair traffic triggered from rack A_i is $t_{i,f}$ ($1 \leq i \neq f \leq r$). We define

TABLE 1
Major Notations Used in this Paper

Notation	Description
k	number of data chunks in a stripe
m	number of parity chunks in a stripe
r	number of racks in a CFS
s	number of stripes associated with the lost chunks
A_i	the i th ($1 \leq i \leq r$) rack
A_f	the rack where the failed node resides ($1 \leq f \leq r$)
λ	load balancing rate
$t_{i,f}$	cross-rack traffic on A_i to repair a failed node in A_f
$c_{i,j}$	number of chunks of the j th stripe in rack A_i
H_i	the i th chunk
H'_i	the i th retrieved chunk for data reconstruction
e	number of iterations in the greedy algorithm for load balancing
u	tolerable number of failed racks ($1 \leq u \leq m$)

the *load balancing rate* λ as the ratio of the maximum amount of cross-rack repair traffic across each rack to the average amount of cross-rack repair traffic over the $(r-1)$ intact racks

$$\lambda = \frac{\max\{t_{i,f} | 1 \leq i \neq f \leq r\}}{\sum_{1 \leq i \neq f \leq r} \frac{t_{i,f}}{r-1}}.$$

Obviously, if there exists cross-rack repair traffic, then $\lambda \geq 1$. Also, we say that the recovery solution is more balanced if its load balancing rate is closer to 1. Therefore, we can formulate the following optimization problem:

$$\text{Minimize } \lambda,$$

subject to

$$\sum_{1 \leq i \neq f \leq r} t_{i,f} \text{ is minimized.}$$

Our optimization goal is to minimize the load balancing rate, subject to the condition that the total amount of cross-rack repair traffic is minimized.

4 CROSS-RACK-AWARE RECOVERY

We thus present CAR, a *cross-rack-aware recovery* algorithm. CAR has three design objectives.

- For each stripe, finding a recovery solution that retrieves chunks from the minimum number of racks.
- Exploiting intra-rack chunk aggregation.
- Exploiting a greedy approach to search for a load-balanced multi-stripe recovery solution.

We justify the design objectives as follows. For each stripe constructed by a (k, m) RS code, any k chunks are sufficient to reconstruct the lost chunk in the stripe. Here, we examine the placement of chunks across racks and identify a recovery solution that retrieves chunks from the minimum number of racks. To repair the lost chunk, instead of directly retrieving and sending individual chunks from a rack, we perform intra-rack chunk aggregation on the retrieved chunks in the same rack and send *one* aggregated chunk (which has the same size as each data/parity chunk) to the replacement node for data reconstruction. Intra-rack chunk aggregation can be realized by separating the reconstruction process of RS codes. By retrieving chunks from the minimum number

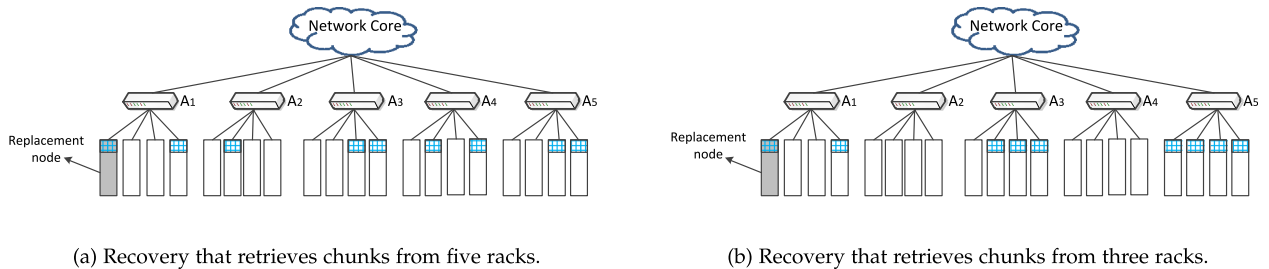


Fig. 3. Two recovery solutions that retrieve data from different sets of racks. Suppose that intra-rack chunk aggregation is performed. To reconstruct the lost chunk of a stripe, for (a), four chunks are transmitted across racks, while for (b), only two chunks are transmitted across racks.

of racks and performing intra-rack chunk aggregation, we minimize the amount of cross-rack repair traffic to reconstruct the lost chunk for each stripe.

For example, suppose that the first node fails in the CFS shown in Fig. 1, which adopts the $(k = 8, m = 6)$ RS code for fault tolerance. Fig. 3 presents two possible recovery solutions, both of which retrieve $k = 8$ chunks yet from a different set of racks to reconstruct the lost chunk of a stripe. By performing intra-rack chunk aggregation, the requested chunks within the same rack will be aggregated into a single chunk. Thus, the recovery solution in Fig. 3a transmits four chunks across racks (i.e., from $A_2, A_3, A_4,$ and A_5), while the one in Fig. 3b only needs to transmit two chunks across racks (i.e., from A_3 and A_5). Note that the retrieval of chunks in A_1 only triggers intra-rack data transmissions, and we assume that it brings limited overhead to the overall recovery performance in a CFS.

In addition, we examine the per-stripe recovery solutions across multiple stripes so as to minimize the load balancing rate. We propose a greedy algorithm that can search for a near-optimal solution with low computational complexity.

4.1 Minimizing the Number of Accessed Racks

We first study how to find a recovery solution that retrieves chunks from the minimum number of racks. Suppose that the lost chunks span s stripes. For the j th stripe ($1 \leq j \leq s$), let $c_{i,j}$ be the number of chunks stored in the i th rack A_i ($1 \leq i \leq r$). Note that we also ensure that the placement of chunks provides rack-level fault tolerance [20]. Here, we assume that we provide single-rack fault tolerance, and we address multi-rack fault tolerance in Section 4.4. For the (k, m) RS code, we require that $c_{i,j} \leq m$, so as to tolerate any single-rack failure; in other words, each stripe should contain at least k chunks in other intact racks of the CFS for data reconstruction.

Suppose that a node fails in rack A_f ($1 \leq f \leq r$). We use $c'_{f,j}$ to denote the number of surviving chunks of the j th stripe ($1 \leq j \leq s$) in A_f in the presence of the node failure. Since every node keeps at most one chunk for a given stripe, we have the following equation:

$$c'_{f,j} = \begin{cases} c_{f,j}, & \text{if } c_{f,j} = 0 \\ c_{f,j} - 1, & \text{if } c_{f,j} \neq 0. \end{cases} \quad (1)$$

Meanwhile, for the remaining $r - 1$ intact racks (i.e., $\{A_1, \dots, A_{f-1}, A_{f+1}, \dots, A_r\}$), they still have the same numbers of chunks in the j th stripe (i.e., $\{c_{1,j}, \dots, c_{f-1,j}, c_{f+1,j}, \dots, c_{r,j}\}$). Given this new setting, Theorem 1 states how

to determine the minimum number of intact racks to be accessed when recovering the lost chunk in the j th stripe ($1 \leq j \leq s$).

Theorem 1. For the j th stripe ($1 \leq j \leq s$), suppose that the numbers of chunks in the $r - 1$ intact racks are ranked in descending order denoted by $\{c_{j_1}, c_{j_2}, \dots, c_{j_{r-1}}\}$, where $c_{j_1} \geq c_{j_2} \geq \dots \geq c_{j_{r-1}}$. We find the smallest number d_j that satisfies

$$c_{j_1} + \dots + c_{j_{d_j}} + c'_{f,j} \geq k. \quad (2)$$

Then d_j is the minimum number of intact racks to be contacted to recover the lost chunk in the j th stripe.

Proof. We prove by contradiction. Suppose that d_j is not the minimum number of intact racks. Let $d'_j < d_j$ be the minimum number of intact racks to be accessed. Then we must have $c_{j_1} + \dots + c_{j_{d'_j}} + c'_{f,j} \geq k$ so that the lost chunk in the j th stripe can be reconstructed. However, this violates our condition that d_j is the minimum value for Equation (2) to be satisfied. \square

We elaborate Theorem 1 via an example. Consider the recovery for the first stripe in the CFS in Fig. 4. The CFS has five racks and employs the $(k = 8, m = 6)$ RS code. For the first stripe, the first rack A_1 originally keeps $c_{1,1} = 4$ chunks. Suppose that the first node in A_1 fails. Then there are $c'_{1,1} = c_{1,1} - 1 = 3$ surviving chunks in A_1 . The numbers of surviving chunks in other four intact racks A_2, A_3, A_4 and A_5 are $c_{2,1} = 1, c_{3,1} = 3, c_{4,1} = 2,$ and $c_{5,1} = 4$, respectively. To reconstruct the lost chunk, we need $k = 8$ surviving chunks for the reconstruction in RS codes. To determine the minimum number of intact racks to be accessed, we first sort the numbers of surviving chunks in the four intact racks, and obtain $(4, 3, 2, 1)$. We can then find $d_1 = 2$, since $4 + 3 + c'_{1,1} = 10 > k = 8$. Thus, we should retrieve the surviving chunks from A_5 and A_3 , as well as the surviving chunks in A_1 , to reconstruct the lost chunk.

We say that a recovery solution is *valid* if it can recover the lost chunk for the j th stripe ($1 \leq j \leq s$) by accessing d_j intact racks only. A valid solution of the j th stripe ($1 \leq j \leq s$) should satisfy the condition that the number of retrieved chunks from d_j intact racks plus the number of surviving chunks in A_f should be no less than k .

We emphasize that a stripe may contain more than one valid recovery solution. We again consider the example of Fig. 4. In addition to the recovery solution that retrieves surviving chunks from A_3 and A_5 , we can also find another recovery solution that retrieves chunks from A_3 and A_4 instead, since $c_{3,1} + c_{4,1} + c'_{1,1} = k = 8$. The latter recovery

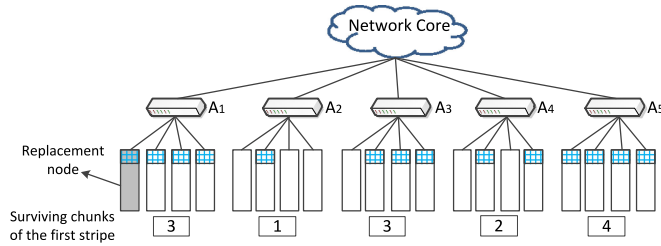


Fig. 4. Example of determining the minimum number of intact racks to be accessed when recovering the lost chunk in the first stripe. Suppose that the CFS employs the $(k = 8, m = 6)$ RS code, and that the first node in A_1 fails. The replacement node can retrieve chunks from the intact racks A_3 and A_5 , as well as from the nodes within the same rack A_1 , for data reconstruction.

solution is also valid, since it can also repair the lost chunk by accessing $d_1 = 2$ intact racks only.

4.2 Intra-Rack Chunk Aggregation

After finding the minimum number of intact racks to be accessed for recovery, we perform intra-rack chunk aggregation on the retrieved chunks in the same rack. We call the aggregation operation *partial decoding*, as it performs part of the decoding steps to reconstruct the lost chunk of a stripe.

To describe how partial decoding works, we first review the encoding and decoding procedures of the (k, m) RS code. Suppose there are k data chunks $\{H_1, H_2, \dots, H_k\}$. Note that most practical storage systems deploy systematic erasure codes (see Section 2.1), meaning that the original data chunks are kept in uncoded form after encoding and hence read requests can directly access the original data. To generate the m parity chunks (denoted by $\{H_{k+1}, \dots, H_{k+m}\}$), the encoding operation can be realized by multiplying a $(k + m) \times k$ matrix $\mathcal{G} = (\mathbf{g}_1 \dots \mathbf{g}_{k+m})^T$ with the k data chunks, i.e.,

$$\begin{pmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_k \\ \vdots \\ \mathbf{g}_{k+m} \end{pmatrix} \cdot \begin{pmatrix} H_1 \\ \vdots \\ H_k \end{pmatrix} = \begin{pmatrix} H_1 \\ \vdots \\ H_k \\ \vdots \\ H_{k+m} \end{pmatrix}. \quad (3)$$

Here, \mathbf{g}_i ($1 \leq i \leq k + m$) is a row vector and its size is $1 \times k$. To make the original data kept in uncoded form, $(\mathbf{g}_1 \dots \mathbf{g}_k)^T$ should be a $k \times k$ identity matrix, where T denotes a matrix or vector transpose operation.

In the decoding operation, RS codes can always use any k surviving chunks (denoted by $\{H'_1, \dots, H'_k\}$) to reconstruct the original data chunks. This implies that there always exists a $k \times k$ invertible matrix \mathcal{X} , such that

$$\mathcal{X} \cdot \begin{pmatrix} H'_1 \\ \vdots \\ H'_k \end{pmatrix} = \begin{pmatrix} H_1 \\ \vdots \\ H_k \end{pmatrix}. \quad (4)$$

Therefore, to reconstruct a chunk H_i ($1 \leq i \leq k + m$), we can derive the following equation based on Equations (3) and (4)

$$H_i = \mathbf{g}_i \cdot \begin{pmatrix} H_1 \\ \vdots \\ H_k \\ \vdots \\ H_{k+m} \end{pmatrix} = \mathbf{g}_i \cdot \mathcal{X} \cdot \begin{pmatrix} H'_1 \\ \vdots \\ H'_k \end{pmatrix}. \quad (5)$$

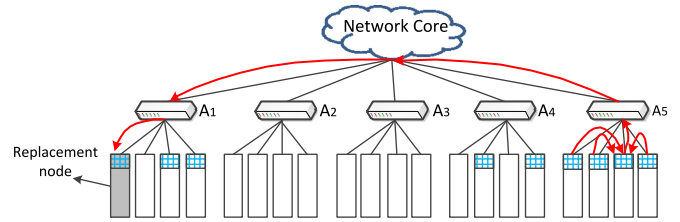


Fig. 5. Example of reconstructing the lost chunk in the first stripe via partial decoding. For example, four chunks in rack A_5 are selected for reconstruction. One node in A_5 performs partial decoding on the four selected chunks and sends the partially decoded chunk to the replacement node.

Let $\mathbf{y} = \mathbf{g}_i \cdot \mathcal{X}$. As the sizes of \mathbf{g}_i and \mathcal{X} are $1 \times k$ and $k \times k$, respectively, $\mathbf{y} = (y_1 \dots y_k)$ is a $1 \times k$ vector. Then we can derive the following equation based on Equation (5)

$$H_i = \mathbf{y}_i \cdot \begin{pmatrix} H'_1 \\ \vdots \\ H'_k \end{pmatrix} = (y_1 \dots y_k) \cdot \begin{pmatrix} H'_1 \\ \vdots \\ H'_k \end{pmatrix}. \quad (6)$$

Equation (6) implies that the reconstruction of H_i is actually realized by the linear operations performed on the k retrieved chunks. Therefore, to mitigate the cross-rack data transmissions for recovery, we can “aggregate” the retrieved chunks in the same rack before performing cross-rack data transmissions. For example, without loss of generality, suppose that the first j requested chunks $\{H'_1, \dots, H'_j\}$ are stored in the same rack. Then we can specify a node in that rack to perform the linear operations based on Equation (6) and obtain the following result:

$$\sum_{i=1}^j y_i H'_i. \quad (7)$$

The aggregation in Equation (7) is called *partial decoding* and the output is referred to as the *partially decoded chunk*, which has the identical size as each data/parity chunk. The partially decoded chunk will then be sent to the replacement node to complete the reconstruction of the lost chunk. The replacement node simply adds all the partially decoded chunks received from A_f and other intact racks that are accessed, in order to reconstruct the lost chunk. We can observe that after applying partial decoding, the amount of cross-rack repair traffic per stripe in CAR equal to the number of partially decoded chunks transmitted from the accessed intact racks, or equivalently, the number of intact racks to be accessed for recovery. Algorithm 1 summarizes the details of recovering the lost chunk of a stripe.

Algorithm 1. Reconstruction for a Stripe

Input: The set of requested chunks $\{H'_1, \dots, H'_k\}$ for recovering the lost chunk of a stripe.

- 1 **for each rack do**
- 2 **if this rack stores requested chunks then**
- 3 Specify a node in this rack to retrieve the requested chunks
- 4 Perform partial decoding on the requested chunks
- 5 Send the partially decoded chunk to the replacement node
- 6 Add the received partially decoded chunks at the replacement node to recover the lost chunk.

Fig. 5 shows an example of how we reconstruct the lost chunk of a stripe via partial decoding. Suppose we need to

retrieve $k = 8$ chunks, and the requested chunks are denoted by $\{H'_1, H'_2, \dots, H'_8\}$ (from left to right). To recover the lost chunk in rack A_1 , we first perform the partial decoding by aggregating the requested chunks in A_1 , A_4 , and A_5 to be $\sum_{i=1}^2 y_i H'_i$, $\sum_{i=3}^4 y_i H'_i$, and $\sum_{i=5}^8 y_i H'_i$, respectively. Then the replacement node reads the three partially decoded chunks to reconstruct the lost chunk. In this example, there are only two chunks transmitted across racks.

4.3 Load Balancing

As stated in Section 4.1, each stripe can have multiple valid per-stripe recovery solutions. Here, we examine the valid per-stripe recovery solutions across multiple stripes, so as to balance the amount of cross-rack repair traffic across the racks (i.e., minimizing the load balancing rate in Section 3). However, enumerating all possible valid per-stripe recovery solutions can be expensive. To elaborate, suppose that we consider the recovery of s stripes, and there are n_j valid recovery solutions for recovering the lost chunk in the j th stripe ($1 \leq j \leq s$). Then the enumeration approach would require $n_1 \times n_2 \times \dots \times n_s$ trials. Depending on the number of valid recovery solutions in each stripe, the enumeration approach can involve a significantly large number of trials.

To mitigate the computation complexity into smaller, we propose a greedy algorithm to search for a near-optimal multi-stripe recovery solution for balancing the amount of cross-rack repair traffic across racks. Having a greedy recovery algorithm enables us to identify recovery solutions *on the fly*, especially under a dynamic environment with constant changing network conditions (e.g., the changing available network bandwidth) [43], [44]. The main idea is to iteratively replace the currently selected multi-stripe recovery solution with another one that introduces a smaller load balancing rate.

Algorithm 2. Greedy Algorithm for Load Balancing

Input: Number of iterations e ; number of stripes s

Output: A multi-stripe recovery solution \mathbb{R} .

```

1 for  $j = 1$  to  $s$  do
2   Select a valid recovery solution  $R_j$  for the  $j$ th stripe
3 Initialize  $\mathbb{R} = \{R_1, R_2, \dots, R_s\}$ 
4 for iteration 1 to  $e$  do
5   Find the intact rack  $A_l$  ( $1 \leq l \neq f \leq r$ ) with the highest  $t_{l,f}$ 
6   for each intact rack  $A_i$  ( $1 \leq i \neq f \leq r$ ) do
7     if  $A_i \neq A_l$  and  $t_{l,f} - t_{i,f} \geq 2$  then
8       Find  $R_j$  and another valid recovery solution  $R'_j$  that
         retrieves no data from  $A_l$  but from  $A_i$  instead
9       if both  $R_j$  and  $R'_j$  exist then
10        Set  $\mathbb{R} = \{R_1, \dots, R_{j-1}, R'_j, R_{j+1}, \dots, R_s\}$ 
11        Jump to the next iteration of the for-loop in step 4
12   Exit the for-loop in step 4 if there is no substitution in  $\mathbb{R}$ 

```

Algorithm 2 shows the details of our greedy algorithm. Suppose that a node in A_f fails ($1 \leq f \leq r$). We first select a valid recovery solution to repair the lost chunk in each stripe, and construct an initial multi-stripe recovery solution \mathbb{R} (steps 1-3). Here, for each stripe, we can follow Theorem 1 to choose the valid recovery solution whose intact racks have the most chunks for the stripe. We then replace the per-stripe recovery solutions in \mathbb{R} over a configurable number of iterations (denoted by e), so as to reduce the load

balance rate. The selection of e depends on both the system's computational capacity and the expected load balancing rate. In general, there are two ways to determine the value of e : (i) the system can choose the maximum value of e that is allowed, subject to the computational capacity constraint; or (ii) the system defines the expected gap of load balancing rates obtained in any two adjacent iterations, such that the algorithm terminates once the reduction of the load balancing rate is lower than the given gap. Specifically, in each iteration, we locate the rack A_l ($1 \leq l \neq f \leq r$) with the highest $t_{l,f}$ (i.e., generating the most cross-rack repair traffic) (steps 4-5). To find a more balanced recovery solution, we scan the remaining intact racks except A_l and select one of the intact racks A_i ($i \neq l$ and $1 \leq i \neq f \leq r$) that satisfies the following condition:

$$t_{l,f} - t_{i,f} \geq 2. \quad (8)$$

Once identifying A_l and A_i , the algorithm scans the current per-stripe recovery solutions in \mathbb{R} . If the per-stripe recovery solution R_j for the j th stripe ($1 \leq j \leq s$) reads chunks from rack A_l , then we check if there exists another valid recovery solution R'_j that can read chunks in A_i , meaning that it can substitute the retrieval from A_l (step 8). If both R_j and R'_j exist, we can substitute R_j with R'_j (steps 9-11). With partial decoding (see Section 4.2), we ensure that we retrieve one less partially decoded chunk from A_l while one more from A_i . Thus, Equation (8) ensures that $t_{l,f} \geq t_{i,f}$ after the substitution, and that the rack with the maximum amount of cross-rack repair traffic generated by a rack is monotonically decreasing. After the substitution, the algorithm resumes another iteration of the for-loop in Step 4 (step 11). If there is no substitution in \mathbb{R} , the algorithm exits the for-loop (step 12). As Algorithm 2 proceeds, the load balancing rate λ of \mathbb{R} iteratively decreases.

Fig. 6 shows an example of how our load balancing scheme works. We consider a CFS that has the same architecture and data layout as in Fig. 1. The CFS also employs the ($k = 8$, $m = 6$) RS code for fault tolerance. For brevity, we only illustrate the chunks retrieved for recovery. Suppose that the first node fails, Fig. 6a first gives an initial multi-stripe recovery solution that recovers the lost chunks of four stripes. With partial decoding, the amount of cross-rack repair traffic can be represented by the number of partially decoded chunks transmitted from each intact rack. For example, A_2 transmits four partially decoded chunks (i.e., $t_{2,1} = 4$) to recover the four lost chunks. Thus, the load balancing rate of the initial recovery solution is $\lambda = \frac{t_{2,1}}{(t_{2,1} + t_{3,1} + t_{4,1} + t_{5,1})/4} = \frac{16}{9}$. Obviously, in Fig. 6a, A_2 (i.e., A_l in Algorithm 2) is the rack with the most cross-rack traffic $t_{2,1} = 4$ (i.e., $t_{l,f}$). To find a more balanced solution, Algorithm 2 locates A_3 that satisfies the condition $t_{2,1} - t_{3,1} = 3 \geq 2$. The algorithm selects the per-stripe recovery solution for the third stripe, such that it retrieves a partially decoded chunk from A_3 instead of A_2 . Fig. 6b shows the new multi-stripe recovery solution. We can see that after the substitution, the load balancing rate of the updated recovery solution is $\lambda = \frac{t_{2,1}}{(t_{2,1} + t_{3,1} + t_{4,1} + t_{5,1})/4} = \frac{12}{9}$, which is smaller than that in Fig. 6a.

Complexity Analysis. We now analyze the complexity of Algorithm 2. In each iteration, the algorithm finds the intact rack with the most cross-rack repair traffic, and search for

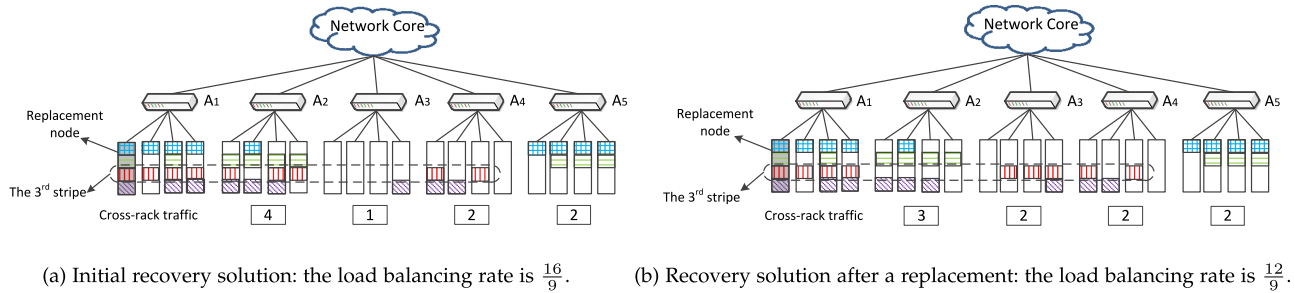


Fig. 6. Example of how to substitute a per-stripe recovery solution in Algorithm 2. The chunks with the same color and fill patterns denote the retrieved chunks for recovery of the same stripe. Compared with the initial multi-stripe recovery solution, the updated multi-stripe recovery solution has a lower load balancing rate, by substituting the per-stripe recovery solution for the third stripe.

another intact rack and per-stripe recovery solution for substitution (steps 6-11). The whole iteration needs no more than $r \times s$ trials. Since the algorithm repeats e iterations, its overall complexity is $O(e \times r \times s)$, which is in polynomial time.

4.4 Multi-Rack Fault Tolerance

We thus far assume that CAR achieves only single-rack fault tolerance, by placing no more than m chunks in each rack (see Section 4.1). We now generalize the rack-level fault tolerance problem and examine how to arrange the chunk placement so that CAR can tolerate multiple rack failures.

Our insight is that using intra-rack chunk aggregation (see Section 4.2), we can reduce more cross-rack repair traffic by placing more chunks of each stripe in a rack. However, to provide rack-level fault tolerance, we must spread the chunks of each stripe across multiple racks. Our goal is to formalize the chunk placement requirement so as to distribute the chunks of each stripe as “compact” as possible (i.e., spanning the least number of racks), while satisfying the required rack-level fault tolerance.

We now define the notation. For a (k, m) RS code, we disperse the $k + m$ chunks of each stripe over r racks, where $1 \leq r \leq k + m$. Suppose that our goal is to tolerate u rack failures, where $1 \leq u \leq \min\{r, m\}$. Theorem 2 states how we place the chunks of a stripe over the minimum number of racks.

Theorem 2. For a (k, m) RS code, we can tolerate the failures of any u out of r racks if and only if the number of racks spanned by a stripe satisfies the following condition:

$$r \geq u + \left\lceil \frac{k}{\lfloor \frac{m}{u} \rfloor} \right\rceil. \quad (9)$$

Proof. We first prove the “Only if” part. Without loss of generality, we rank the numbers of chunks of the j th stripe in the r racks in descending order denoted by $\{c_{j_1}, c_{j_2}, \dots, c_{j_r}\}$, where $c_{j_1} \geq c_{j_2} \geq \dots \geq c_{j_r}$. To tolerate any u rack failures, we require that

$$c_{j_1} + c_{j_2} + \dots + c_{j_u} \leq m. \quad (10)$$

Equation (10) states that we must store no more than m chunks in any u racks for fault tolerance. From Equation (10), we can also show the following:

$$c_{j_u} \leq \frac{c_{j_1} + c_{j_2} + \dots + c_{j_u}}{u} \leq \left\lfloor \frac{m}{u} \right\rfloor. \quad (11)$$

Suppose that the u racks that store the most chunks of the j th stripe now all fail. The remaining $k + m - \sum_{i=1}^u c_{j_i}$ chunks are stored in the remaining $r - u$ surviving racks. Since each surviving rack stores no more than c_{j_u} chunks, the number of surviving racks $r - u$ must satisfy

$$r - u \geq \left\lceil \frac{k + m - \sum_{i=1}^u c_{j_i}}{c_{j_u}} \right\rceil \geq \left\lceil \frac{k}{\lfloor \frac{m}{u} \rfloor} \right\rceil, \quad (12)$$

due to Equations (10) and (11). The “Only if” part holds.

We now prove the “If” part. The intuition is to find a chunk placement that satisfies Equation (9). We fix $r = r^*$, where $r^* = u + \lceil \frac{k}{\lfloor \frac{m}{u} \rfloor} \rceil$. To place $k + m$ chunks of the j th stripe over r^* racks, we set $c_{j_2} = c_{j_3} = \dots = c_{j_{r^*-1}} = \lfloor \frac{m}{u} \rfloor$, $c_{j_1} = m - (c_{j_2} + c_{j_3} + \dots + c_{j_u})$, and $c_{j_{r^*}} = k - (c_{j_{u+1}} + \dots + c_{j_{r^*-1}})$. We can easily verify that c_{j_1} is the largest, $c_{j_{r^*}}$ is the smallest and non-negative, and hence the list $c_{j_1}, c_{j_2}, \dots, c_{j_{r^*}}$ are in descending order. Now, we can show that $c_{j_1} + c_{j_2} + \dots + c_{j_u} \leq m$, and hence for every u out of r^* racks, there must be no more than m chunks. Thus, this chunk placement can tolerate any u rack failures. The “If” part holds. \square

For example, suppose that we deploy a $(k = 7, m = 5)$ RS code and want to tolerate any $u = 2$ rack failures. Then we should choose $r^* = 2 + \lceil \frac{7}{\lfloor \frac{5}{2} \rfloor} \rceil = 6$ racks. We set $c_{j_2} = c_{j_3} = \dots = c_{j_{r^*-1}} = \lfloor \frac{5}{2} \rfloor = 2$, $c_{j_1} = 5 - 2 = 3$, and $c_{j_{r^*}} = 7 - (2 + 2 + 2) = 1$, such that the distribution of chunks is $(3, 2, 2, 2, 2, 1)$. We can verify that this chunk placement can tolerate any $u = 2$ rack failures.

4.5 Analysis

We conduct analysis on the design implications of CAR under different settings.

4.5.1 Impact of Cross-Rack Bandwidth

We first analyze how the recovery time of CAR varies with the cross-rack bandwidth. Let C be the chunk size and B be the available cross-rack bandwidth. Here, we assume that the recovery time is mostly dominated by the cross-rack transfer time instead of by other factors, such as CPU encoding/decoding time and intra-rack transfer time. Suppose that CAR retrieves one chunk from each of other r' racks through intra-rack chunk aggregation (where $r' < r$). The recovery time of CAR is $\frac{r'C}{B}$. To illustrate the implication, suppose that all nodes are interconnected by a 1 Gb/s

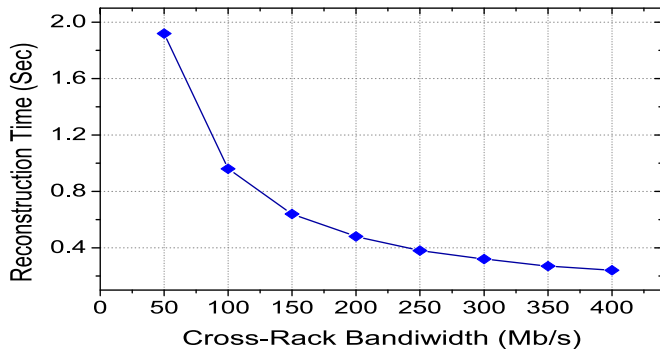


Fig. 7. The reconstruction time when the cross-rack bandwidth varies.

Ethernet, and that the over-subscription ratio ranges from 2.5 to 20 (i.e., the cross-rack bandwidth ranges from 50 to 400 Mb/s). Fig. 7 plots the recovery time versus the cross-rack bandwidth. We can observe that the recovery time increases when the over-subscription ratio becomes larger (i.e., the cross-rack bandwidth is smaller).

4.5.2 Impact of Rack Fault Tolerance

We also analyze how CAR makes a design trade-off between the amount of cross-rack recovery traffic and the number of rack failures that can be tolerated (i.e., u). We select a ($k = 10, m = 8$) erasure code and assume that the size of a chunk is 4 MB. Given an expected number of tolerated rack failures, we first derive a chunk distribution based on the method in Section 4.4. We then measure the average amount of cross-rack repair traffic caused by CAR to recover each chunk. The results are shown in Fig. 8, which indicates that the amount of cross-rack repair traffic increases as the number of tolerable rack failures increases.

Discussion. When under the same chunk distribution and failure rates, CAR needs less recovery time than the random recovery (i.e., randomly select k surviving chunks and directly send them to the replacement node, see Section 5 for details), and makes the system stay at the reliable state for longer time. Therefore, CAR can improve the system reliability when compared with traditional random recovery.

4.6 Extension of CAR

CAR mainly focuses on node recovery based on the network topology of CFSes. Nevertheless, the design principle of CAR can still provide a valuable reference for node recovery for general network topologies, provided that over-subscription exists. Specifically, if the recovery has to deliver the requested data to the replacement node over a bandwidth-limited connection, then the system can first perform partial decoding (as in CAR) and send the partial decoded result to the replacement node, so as to mitigate the congestion on the bandwidth-limited connection and reduce the overall recovery time.

Currently, CAR mainly focuses on RS code, and we accordingly design three techniques for CAR. In fact, the first two techniques (i.e., minimizing the number of accessed racks and intra-rack chunk aggregation) can also help to reduce the cross-rack data transfer if we use XOR-based erasure codes

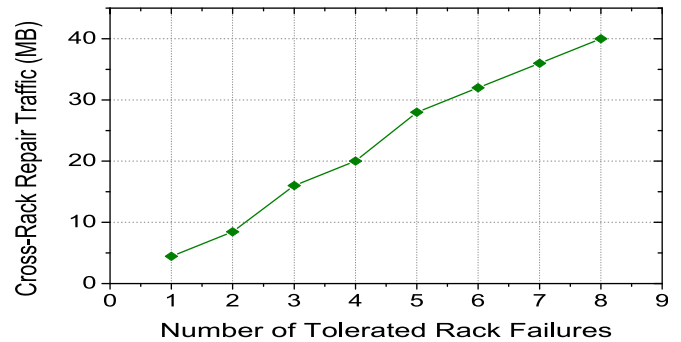


Fig. 8. The amount of cross-rack recovery traffic when the number of tolerated rack failures varies.

(see Section 2.2). We can first find the minimum number of accessed racks given the distribution of surviving chunks and the decoding rules of a specific XOR-based erasure code, followed by aggregating the data within each rack based on XOR operations. For load balancing, we may need a different greedy algorithm for XOR-based erasure codes; we pose it as future work.

5 PERFORMANCE EVALUATION

5.1 Implementation Overview

We implement a prototype of CAR in C on Linux. We implement RS codes, whose encoding and decoding operations are realized based on the open-source library Jersure 1.2. [27]. We configure the size of a chunk in CAR ranging from 4 to 16 MB. In failure recovery, each selected chunk will be partitioned into many *sub-chunks* whose sizes are the order of kilobytes (e.g., 4 KB), and the recovery operation is performed in a pipelined manner. In each rack, we select one node to perform partial decoding. Each other node (except the replacement node and the partial decoding node) will establish a socket connection and transfers a sub-chunk to the partial decoding node, which uses *aio* library [3] to perform asynchronous reads to collect the sub-chunks. After that, the partial decoding node computes the partial decoded sub-chunk based on the received sub-chunks and sends it to the replacement node via a socket connection. The replacement node will recover a sub-chunk after adding all the partial decoded sub-chunks. We repeat the procedure for all sub-chunks until the lost chunk is successfully repaired.

5.2 Evaluation Results

We conduct extensive testbed experiments to evaluate the performance of CAR. We would like to answer the following questions:

- 1) How much cross-rack traffic and the time of single failure recovery can be reduced by CAR?
- 2) Will CAR sustain its effectiveness when deployed over different CFS configurations, including the number of racks, the number of nodes per rack, and the erasure code parameters?
- 3) How do the iteration steps affect the load balancing rate?
- 4) Will CAR increase the computational overhead for recovery?

TABLE 2
Configurations of Three CFS Settings

CFSes	A_1	A_2	A_3	A_4	A_5	RS code
CFS1	4	3	3			$k = 4, m = 3$
CFS2	4	3	3	3		$k = 6, m = 3$
CFS3	6	4	5	3	2	$k = 10, m = 4$

*Evaluation Environment.*¹ We conduct our evaluation on three CFS settings with different architectures and RS code parameters. Table 2 shows the configurations of the CFS settings for our evaluation, including the selected RS codes and the number of nodes in each rack. For example, CFS1 is deployed over three racks with 10 nodes and it selects the ($k = 4, m = 3$) RS code. In practical deployment, even a CFS contains a large number of nodes, the erasure coding parameters are often configured to make $k + m$ not to be too large so as to limit the encoding overhead and the amount of repair traffic, while maintaining fault tolerance [26]. For example, Google Colossus FS [1], [4] uses the ($k = 6, m = 3$) RS Code and HDFS-RAID [1], [4] uses the ($k = 10, m = 4$) RS Code. Thus, each stripe with $k + m$ chunks will only span a limited number of nodes in real-world storage systems, while multiple stripes are independently encoded and repaired under failures. We configure the stripe size $k + m$ in our evaluation to range from 7 to 14, such that this range covers typical system configurations of existing storage systems [1], [4]. Thus, we expect that our CFS configurations are sufficiently practical to reflect the repair performance in real-world deployment.

Table 3 also lists the hardware configurations of the nodes in different racks. We configure the nodes in the same rack to have the same hardware configurations. The racks are connected by the TP-LINK TL-SG1016D 16-Port Gigabit Ethernet switches.

Methodology. We construct 100 stripes and randomly distribute the data and parity chunks of each stripe across all nodes in each CFS, while ensuring single-rack fault tolerance (see Section 4.1). To evaluate the recovery performance, we randomly select a node to erase its stored chunks. We use the same node as the replacement node, and trigger the recovery operation. We apply CAR to find the recovery solution and recover the lost chunk of each stripe. For comparisons, we also consider a baseline approach called *random recovery* (RR), which finds the recovery solution by randomly choosing k surviving chunks of a stripe and sending them to the replacement node for recovery. To start recovery, the replacement node first contacts k surviving nodes for each stripe to simultaneously launch the transmissions of the chunks. For CAR, the replacement node also selects a node in each rack to perform partial decoding, such that the surviving nodes first send their chunks to the selected node in each rack for partial decoding, and then the selected node in each rack sends the aggregated chunk to the replacement node. On the other hand, for RR, the k surviving nodes directly send the chunks to the replacement node. Each of our results is averaged over multiple trials (generally 5 trials to 10 trials). We find that the

¹ Note that we rerun all our experiments and hence our performance results is slightly different from those in our conference version [36], although the key conclusions remain the same.

TABLE 3
Configurations of Nodes in Each Rack

Servers	CPU	Memory	OS	Disk
Nodes in A_1	AMD Opteron (tm) 800 MHz 2,378 Quad-Core processors	16 GB	Fedora 11	1 TB
Nodes in A_2	an Intel Xeon X5472 3.00 GHz Quad-Core CPU	8 GB	SUSE Linux Enterprise Server 11	4 TB
Nodes in A_3	an Intel Xeon E5506 2.13 GHz Quad-Core CPU	8 GB	Fedora 10	1 TB
Nodes in A_4	an Intel Xeon E5420 2.50 GHz Quad-Core CPU	4 GB	Fedora 10	300 GB
Nodes in A_5	an Intel Xeon X5472 3 GHz Quad-Core CPU	8 GB	Ubuntu 10.04.3 LTS	4 TB

standard deviation is small, so we do not plot the standard deviation in the figures.

Experiment 1 (Recovery Performance in Different CFS Settings). We first evaluate the amounts of cross-rack repair traffic due to CAR and RR when recovering a single lost chunk. We conduct the evaluation in the three CFS settings shown in Table 2. Fig. 9 shows the results of cross-rack traffic versus the chunk size. We make the following observations.

In all cases, CAR significantly reduces the amount of cross-rack repair traffic when compared to RR. For example, when the chunk size is 4 MB, CAR can reduce 52.4 percent of cross-rack repair traffic in CFS1 (see Fig. 9a). The reason is that CAR not only finds the recovery solution that involves the minimum number of racks, but also performs partial decoding in each rack before cross-rack data transmissions. Both techniques guarantee the minimum amount of cross-rack data transmissions when reconstructing the lost chunk in each stripe. As a comparison, RR simply retrieves the chunks from other surviving nodes to the replacement node, thereby triggering a considerable amount of cross-rack repair traffic.

In addition, the performance gain of CAR is influenced by the parameter k used in RS codes. In general, when the number of racks is fixed, CAR can reduce more cross-rack data transmissions when k increases. The reason is that in RR, the number of retrieved chunks increases when k becomes larger. On the other hand, CAR ensures that each rack only needs to send one chunk across racks under partial decoding. For example, when the chunk size is 16 MB, the saving of cross-rack repair traffic due to CAR increases to 66.9 percent in CFS3 (see Fig. 9c).

We further compare CAR and RR in terms of the recovery time per lost chunk in different CFS settings. We measure the overall duration starting from the time when all surviving nodes send the chunks until the time when all lost chunks are completely reconstructed. We divide the overall duration by the number of lost chunks being reconstructed to obtain the recovery time per lost chunk.

Fig. 10 shows the recovery time per lost chunk versus the chunk size. It indicates that CAR greatly reduces the recovery time when compared to RR. For example, when the chunk size is 8 MB, to recover a lost chunk in CFS2, CAR

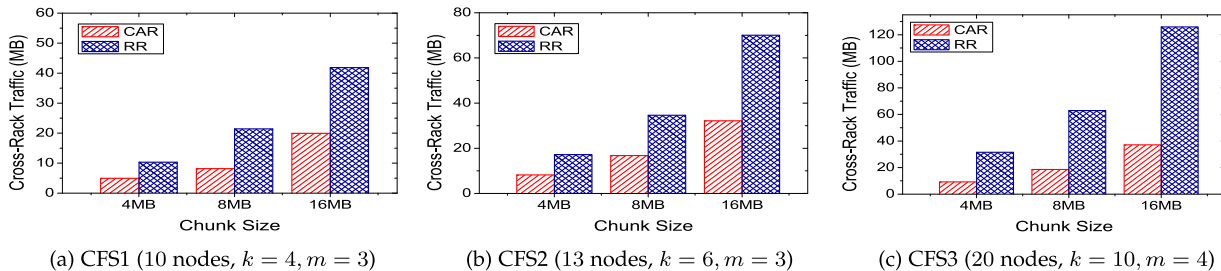


Fig. 9. Experiment 1: Comparisons of the amounts of cross-rack traffic between CAR and RR over different CFSes.

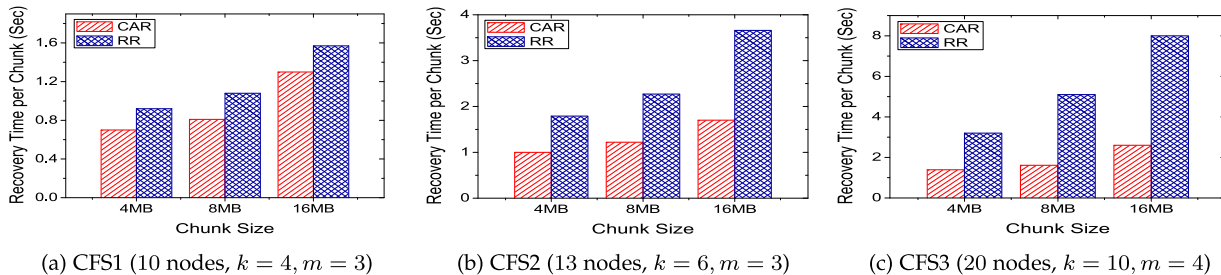


Fig. 10. Experiment 1: Comparisons of recovery times between CAR and RR over different CFSes.

reduces 46.2 percent of recovery time (see Fig. 10b). The reasons are three-fold. First, CAR reduces the amount of cross-rack repair traffic. Second, CAR balances the amount of cross-rack repair traffic across multiple racks, while RR randomly selects k surviving chunks to recover a lost chunk and hence leads to an uneven distribution of cross-rack repair traffic in general. Third, CAR offloads the recovery process to a node in each rack due to partial decoding, while RR requires the replacement node to perform the whole recovery process for all lost chunks.

Experiment 2 (Impact of Number of Racks). We next evaluate the performance of CAR when it is deployed over different number of racks. We consider a new CFS setting as follows. We select the $(k=6, m=3)$ RS code and perform the evaluation when the number of racks increases from three to five, in which we use $\{A_1, A_2, A_3\}$, $\{A_1, A_2, A_3, A_4\}$, and $\{A_1, A_2, A_3, A_4, A_5\}$ in Table 3, respectively. Each rack consists of three nodes, and the nodes in the same rack have the same hardware configurations and operating system (see Table 3). The chunk size is set as 4 MB. In the evaluation, we erase the data on each node and invoke failure recovery. We measure the average amounts of cross-rack repair traffic and recovery time to reconstruct a lost chunk. Fig. 11 shows the evaluation results.

Fig. 11a first presents the amount of cross-rack traffic to repair a lost chunk under different number of racks. We make three observations. First, there will be more cross-rack traffic in CAR when the number of racks increases. The reason is that the distribution of chunks will be more “sparse”

(i.e., fewer chunks in a rack) if they are dispersed across more racks, which will limit the effectiveness of intra-rack chunk aggregation. Second, RR will also introduce more cross-rack repair traffic when the number of rack increases. As shown before, RR needs to directly read any k chunks to repair a lost chunk. Increasing the number of racks will place more chunks in the intact racks, and hence produce more cross-rack repair traffic. Third, CAR still keeps its effectiveness even when the number of racks varies. For example, compared with RR, CAR reduces about 55.3 percent of cross-rack repair traffic when the CFS has three racks. This saving will be 54.9 percent when the number of racks increases to five.

Fig. 11b presents the average recovery time to repair a chunk versus the number of racks. We see that both CAR and RR incurs more recovery time when there are more racks. For example, CAR needs 0.60 seconds to repair a lost chunk when there are three racks, and the recovery time increases to 1.04 seconds when the number of racks is five. CAR still sees performance gain; for example, its recovery time is 33.7 percent less than that of RR when there are five racks.

Experiment 3 (Impact of Erasure Coding Configurations). We further investigate the impact of selected parameters in erasure coding schemes. We consider a new CFS setting as follows. We configure a CFS over four racks $\{A_1, A_2, A_3, A_4\}$, each of which includes three nodes with configurations shown in Table 3. We set the chunk size to be 4 MB and select the $(k=6, m=3)$ RS code. We then evaluate the cross-rack traffic and the time to repair a lost chunk when k

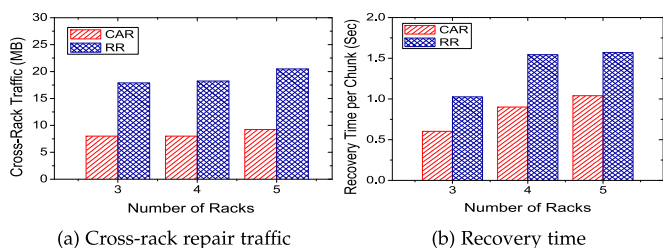


Fig. 11. Experiment 2: Impact of number of racks.

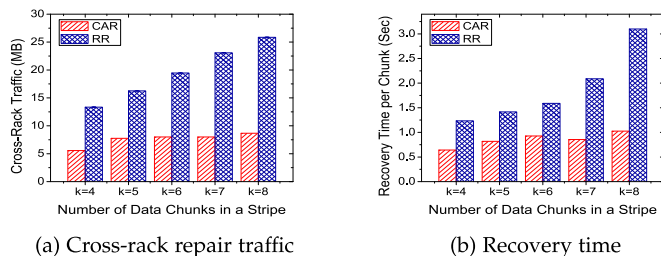


Fig. 12. Experiment 3: Impact of number of data chunks.

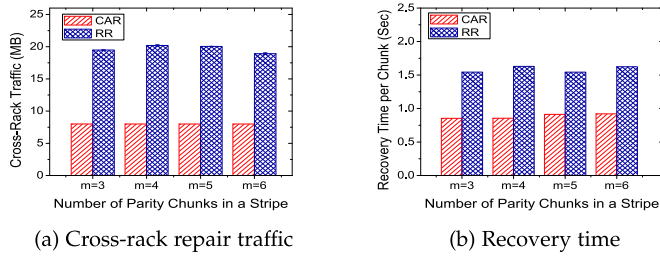


Fig. 13. Experiment 3: Impact of number of parity chunks.

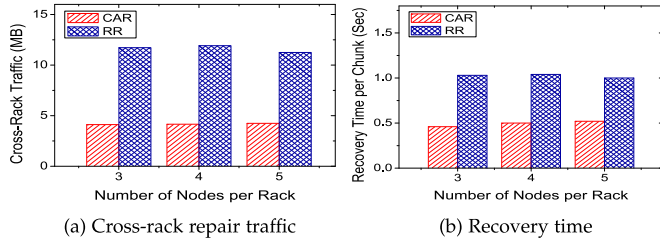


Fig. 14. Experiment 4: Impact of number of nodes per rack.

and m vary respectively. The evaluation results are respectively shown in Figs. 12 and 13.

Fig. 12a first gives the amount of caused cross-rack repair traffic when the number of data chunks (i.e., k) in a stripe varies. We make three observations. First, both CAR and RR incur more cross-rack repair traffic when the value of k becomes larger. The reason is that to repair a lost chunk, k surviving chunks are required in RS Codes. Second, CAR is more insensitive with the change of k when compared to RR. Third, CAR can still reduce the amount of cross-rack repair traffic when the value of k varies.

Fig. 12b shows the time to repair a lost chunk when the value of k varies. We make two observations. First, both of CAR and RR incur more time to reconstruct a chunk when the value of k is larger, mainly because of the increased amount of cross-rack traffic during recovery. Second, CAR incurs less recovery time compared to RR. For example, when $k = 4$, CAR requires about 47.9 percent less time to repair a chunk when compared with RR.

Fig. 13a shows the amount of cross-rack repair traffic when the number of parity chunks (i.e., m) in a stripe changes. We see that the selection of m in our evaluation does not have significant impact on the amount of cross-rack repair traffic for both RR and CAR. The reason is that the configuration of m does not affect the number of chunks to be read for recovery, even though it may increase the number of chunks in a stripe. Following the same reason,

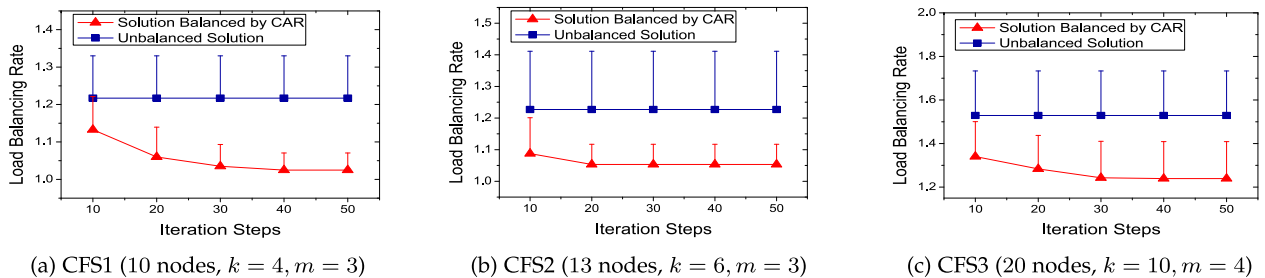
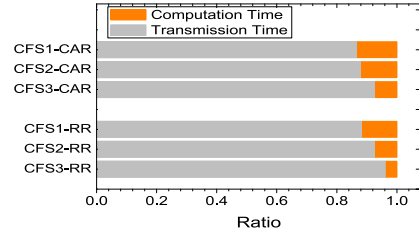
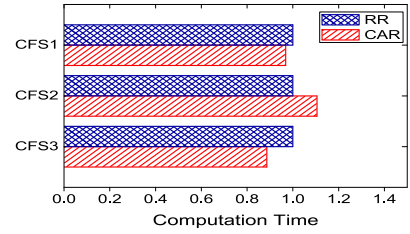


Fig. 15. Experiment 5: Load balancing rate (and the standard deviation) versus the number of iteration steps in CAR. For brevity, we only show the standard deviations in the positive direction.



(a) Ratios of transmission time and computation time



(b) Computation time (normalized with respect to that of RR)

Fig. 16. Evaluation of transmission time and computation time for recovering a lost chunk.

Fig. 13b also indicates that the number of parity chunks has limited influence on the recovery time.

Experiment 4 (Impact of Number of Nodes Per Rack). We further study the impact of number of nodes per rack. We select the ($k = 4, m = 3$) RS code and fix the chunk size as 4 MB. The evaluated CFS in this test is constructed over three racks $\{A_1, A_2, A_3\}$ in Table 3, where each rack has three nodes. We then erase the data on a randomly selected node, vary the number of nodes in each rack from three to five, and measure the average amounts of cross-rack traffic and recovery time to repair a lost chunk. The evaluation results are shown in Fig. 14.

Fig. 14a shows that the amounts of cross-rack repair traffic to repair a lost chunk in both CAR and RR will not be significantly affected when the number of nodes per rack varies. The reason is that each rack always has the same number of nodes (i.e., from three to five) in each test, and this results in the same likelihood of placing a chunk in any one of the racks during the distribution of chunks. As a result, the number of chunks in each rack will not be affected even when the number of nodes per rack varies. Combined with the recovery principle of RR and CAR, their amounts of cross-rack repair traffic will not be influenced.

Fig. 14b shows the recovery time of CAR and RR versus the number of nodes per rack. As the recovery time is closely related to the amount of cross-rack repair traffic, we

observe that the recovery time of both CAR and RR will be stable when the number of nodes per rack changes.

Experiment 5 (Load Balancing). In this evaluation, we measure the capability of CAR to balance the amount of cross-rack repair traffic across multiple racks. We configure the number of iterations (i.e., e) to be 50 and the number of stripes (i.e., s) to be 100 in Algorithm 2. In each CFS setting, we measure the load balancing rate (i.e., λ) of CAR after each number of iterations.

Fig. 15 presents the average results and the standard deviations for CAR with and without performing load balancing (the latter means that we do not execute Algorithm 2). In all cases, CAR can effectively balance the amount of cross-rack repair traffic. For example, in CFS1 (see Fig. 15a), if we do not perform load balancing, the load balancing rate is 1.22 even though CAR retrieves chunks from the minimum number of racks and performs partial decoding. With load balancing enabled, the load balancing rate of the optimized solution can reduce to 1.02. In addition, as we increase the number of iterations, the load balancing rate first decreases significantly and then becomes stable, mainly because the resulting solution is closer to the minimum with the increase of iteration steps.

Experiment 6 (Computation Time and Transmission Time). We further provide a breakdown on the recovery time, in terms of the transmission time and the computation time to recover a lost chunk. The transmission time records the duration of data transmissions over the CFS, while the computation time records the duration to perform required decoding operations over finite fields for reconstructing the lost chunk. We fix the chunk size as 8 MB.

Fig. 16 presents the results. Fig. 16a shows that the transmission time dominates the overall recovery time, justifying the need of reducing the transmission overhead in CAR. Also, the ratio of computation time in both RR and CAR decreases when the parameter k in RS codes increases. For example, for CAR in CFS1 (where $k = 4$), the computation time occupies 11.3 percent of recovery time, while in CFS3, the ratio decreases to 7.1 percent (where $k = 10$).

Fig. 16b shows that the computation time of CAR normalized over that of RR. The computation times of both CAR and RR are similar (e.g., with up to around 10 percent of difference). Note that CAR does not change the decoding operations in RS codes, but only breaks down a decoding operation into multiple partial decoding operations.

6 CONCLUSION

This paper reconsiders the single failure recovery problem in a clustered file system with over-subscribed cross-rack bandwidth, and propose CAR, a *cross-rack-aware recovery* algorithm. CAR includes three key techniques. First, CAR examines the data layout in a CFS and determines the recovery solution that accesses the minimum number of racks for each stripe. Second, CAR performs partial decoding by aggregating the requested chunks in the same rack before cross-rack data transmissions. Third, CAR uses a greedy algorithm to find the recovery solution that balances the amount of cross-rack repair traffic across racks. Experimental results show that CAR can reduce both cross-rack data transmissions and the overall recovery time.

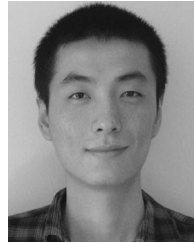
ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 61602120, 61672159, 61232003, 61433008), the Technology Innovation Platform Project of Fujian Province (Grant No. 2014H2005), the Fujian Collaborative Innovation Center for Big Data Application in Governments, the Fujian Engineering Research Center of Big Data Analysis and Processing, and the Fujian Provincial Natural Science Foundation (Grant No. 2017J05102). This work is also supported by the Research Grants Council of Hong Kong (GRF 14216316 and CRF C7036-15G). An earlier version of this paper appeared at the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'16) [36]. In this journal version, we include additional analysis on rack-level fault tolerance and more evaluation results for CAR.

REFERENCES

- [1] Colossus, successor to google file system. (2012). [Online]. Available: http://static.googleusercontent.com/media/research.google.com/en/us/university/relations/facultysummit2010/storage_architecture_and_challenges.pdf
- [2] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.
- [3] S. Bhattacharya, S. Pratt, B. Pulavarty, and J. Morgan, "Asynchronous I/O support in Linux 2.5," in *Proc. Linux Symp.*, 2003, pp. 371–386.
- [4] D. Borthakur, R. Schmidt, R. Vadali, S. Chen, and P. Kling, "HDFS RAID," in *Proc. Hadoop User Group Meet.*, 2010.
- [5] B. Calder, et al., "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. ACM Symp. Operating Syst. Principles*, 2011, pp. 143–157.
- [6] M. Chowdhury, S. Kandula, and I. Stoica, "Leveraging endpoint flexibility in data-intensive clusters," in *Proc. ACM SIGCOMM*, 2013, pp. 231–242.
- [7] P. Corbett, et al., "Row-diagonal parity for double disk failure correction," in *Proc. USENIX Conf. File Storage Technol.*, 2004, pp. 1–1.
- [8] J. Dean, "Software engineering advice from building large-scale distributed systems," CS295 Lecture at Stanford University, Stanford, CA, USA, Jul. 2007.
- [9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. 6th Conf. Symp. Operating Syst. Des. Implementation*, 2004, pp. 10–10.
- [10] A. G. Dimakis, P. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.
- [11] D. Ford, et al., "Availability in globally distributed storage systems," in *Proc. USENIX Symp. Operating Syst. Des. Implementation*, 2010, pp. 61–74.
- [12] Y. Fu, J. Shu, and X. Luo, "A stack-based single disk failure recovery scheme for erasure coded storage systems," in *Proc. IEEE Int. Symp. Reliable Distrib. Syst.*, 2014, pp. 136–145.
- [13] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. ACM Symp. Operating Syst. Principles*, 2003, pp. 29–43.
- [14] Y. Hu, P. P. Lee, and X. Zhang, "Double regenerating codes for hierarchical data centers," in *Proc. IEEE Int. Symp. Inf. Theory*, 2016, pp. 245–249.
- [15] Y. Hu, et al., "Optimal repair layering for erasure-coded data centers: From theory to practice," *ACM Trans. Storage*, vol. 13, 2017, Art. no. 33.
- [16] C. Huang, et al., "Erasure coding in windows azure storage," in *Proc. USENIX Conf. Annu. Tech. Conf.*, 2012, pp. 2–2.
- [17] C. Huang and L. Xu, "STAR: An efficient coding scheme for correcting triple storage node failures," *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 889–901, Jul. 2008.
- [18] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads," in *Proc. USENIX Conf. File Storage Technol.*, 2012, pp. 20–20.

- [19] M. Li and P. P. Lee, "STAIR codes: A general family of erasure codes for tolerating device and sector failures in practical storage systems," in *Proc. USENIX Conf. File Storage Technol.*, 2014, pp. 147–162.
- [20] R. Li, Y. Hu, and P. P. Lee, "Enabling efficient and reliable transition from replication to erasure coding for clustered file systems," in *Proc. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2015, pp. 148–159.
- [21] R. Li, X. Li, P. P. Lee, and Q. Huang, "Repair pipelining for erasure-coded storage," in *Proc. USENIX Annu. Tech. Conf.*, 2017, pp. 567–579.
- [22] X. Luo and J. Shu, "Load-balanced recovery schemes for single-disk failure in storage systems with any erasure code," in *Proc. IEEE Int. Conf. Parallel Process.*, 2013, pp. 552–561.
- [23] S. Mitra, R. Panta, M.-R. Ra, and S. Bagchi, "Partial-parallel-repair (PPR): A distributed technique for repairing erasure coded storage," in *Proc. 11th Eur. Conf. Comput. Syst.*, 2016, Art. no. 30.
- [24] S. Muralidhar, et al., "F4: Facebooks warm BLOB storage system," in *Proc. USENIX Conf. Operating Syst. Des. Implementation*, 2014, pp. 383–398.
- [25] J. S. Plank, M. Blaum, and J. L. Hafner, "SD codes: Erasure codes designed for how storage systems really fail," in *Proc. USENIX Conf. File Storage Technol.*, 2013, pp. 95–104.
- [26] J. S. Plank, J. Luo, C. D. Schuman, L. Xu, and Z. Wilcox-O'Hearn, "A performance evaluation and examination of open-source erasure coding libraries for storage," in *Proc. USENIX Conf. File Storage Technol.*, 2009, pp. 253–265.
- [27] J. S. Plank, S. Simmerman, and C. D. Schuman, "Jerasure: A library in C/C++ facilitating erasure coding for storage applications-version 1.2," Dept. Elect. Eng. Comput. Sci., Univ. Tennessee, Knoxville, TN, USA, Tech. Rep. CS-08-627, 2008.
- [28] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for I/O, storage, and network-bandwidth," in *Proc. USENIX Conf. File Storage Technol.*, 2015, pp. 81–94.
- [29] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers," in *Proc. ACM SIGCOMM*, 2014, pp. 331–342.
- [30] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.
- [31] M. Sathiamoorthy, M. Asteris, D. Papailopoulos, and A. E. A. Dimakis, "XORing elephants: Novel erasure codes for big data," *Proc. VLDB Endowment*, vol. 6, pp. 325–336, 2013.
- [32] F. B. Schmuck and R. L. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proc. USENIX Conf. File Storage Technol.*, 2002, Art. no. 19.
- [33] B. Schroeder and G. Gibson, "Disk failures in the real world: What does an MTTf of 1,000,000 hours mean to you?" in *Proc. USENIX Conf. File Storage Technol.*, 2007, Art. no. 1.
- [34] Z. Shen and J. Shu, "HV code: An all-around MDS code to improve efficiency and reliability of RAID-6 systems," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2014, pp. 550–561.
- [35] Z. Shen, J. Shu, and Y. Fu, "Seek-efficient I/O optimization in single failure recovery for XOR-coded storage systems," in *Proc. IEEE 34th Symp. Reliable Distrib. Syst.*, 2015, pp. 228–237.
- [36] Z. Shen, J. Shu, and P. P. C. Lee, "Reconsidering single failure recovery in clustered file systems," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2016, pp. 323–334.
- [37] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proc. IEEE Symp. Mass Storage Syst. Technol.*, 2010, pp. 1–10.
- [38] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *Proc. Revised Papers 1st Int. Workshop Peer-to-Peer Syst.*, 2002, pp. 328–338.
- [39] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, "A tale of two erasure codes in HDFS," in *Proc. USENIX Conf. File Storage Technol.*, 2015, pp. 213–226.
- [40] L. Xiang, Y. Xu, J. Lui, and Q. Chang, "Optimal recovery of single disk failure in RDP code storage systems," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 2010, pp. 119–130.
- [41] L. Xu and J. Bruck, "X-code: MDS array codes with optimal encoding," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 272–276, Jan. 1999.
- [42] S. Xu, et al., "Single disk failure recovery for X-code-based parallel storage systems," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 995–1007, Apr. 2014.
- [43] Y. Zhu, P. P. Lee, Y. Hu, L. Xiang, and Y. Xu, "On the speedup of single-disk failure recovery in XOR-coded storage systems: Theory and practice," in *Proc. IEEE Symp. Mass Storage Syst. Technol.*, 2012, pp. 1–12.
- [44] Y. Zhu, P. P. Lee, L. Xiang, Y. Xu, and L. Gao, "A cost-based heterogeneous recovery scheme for distributed storage systems with RAID-6 codes," in *Proc. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2012, pp. 1–12.



Zhirong Shen received the BS degree from the University of Electronic Science and Technology of China and the PhD degree from the Department of Computer Science and Technology, Tsinghua University, in 2016. He is now a post-doctoral fellow with the Chinese University of Hong Kong. His current research interests include storage reliability and storage security.



Patrick P. C. Lee received the BEng (first-class honors) degree in information engineering from the Chinese University of Hong Kong, in 2001, the MPhil degree in computer science and engineering from the Chinese University of Hong Kong, in 2003, and the PhD degree in computer science from Columbia University, in 2008. He is now an associate professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong. His research interests include cloud storage, distributed systems and networks, and security/resilience.



Jiwu Shu received the PhD degree in computer science from Nanjing University, in 1998, and finished the postdoctoral position research with Tsinghua University, in 2000. Since then, he has been teaching with Tsinghua University. His current research interests include storage security and reliability, non-volatile memory based storage systems, and parallel and distributed computing. He is a senior member of the IEEE.



Wenzhong Guo received the BS and MS degrees in computer science, and the PhD degree in communication and information system from Fuzhou University, Fuzhou, China, in 2000, 2003, and 2010, respectively. He is currently a full professor with the College of Mathematics and Computer Science, Fuzhou University. His research interests include intelligent information processing, sensor networks, network computing, and network performance evaluation.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

EC2: Ensemble Clustering and Classification for Predicting Android Malware Families

Tanmoy Chakraborty¹, Fabio Pierazzi, and V. S. Subrahmanian

Abstract—As the most widely used mobile platform, Android is also the biggest target for mobile malware. Given the increasing number of Android malware variants, detecting *malware families* is crucial so that security analysts can identify situations where signatures of a known malware family can be adapted as opposed to manually inspecting behavior of all samples. We present EC2 (Ensemble Clustering and Classification), a novel algorithm for discovering Android malware families of *varying sizes*—ranging from very large to very small families (even if previously unseen). We present a performance comparison of several traditional classification and clustering algorithms for Android malware family identification on DREBIN, the largest public Android malware dataset with labeled families. We use the output of both supervised classifiers and unsupervised clustering to design EC2. Experimental results on both the DREBIN and the more recent Koodous malware datasets show that EC2 accurately detects both small and large families, outperforming several comparative baselines. Furthermore, we show how to automatically characterize and explain unique behaviors of specific malware families, such as FakeInstaller, MobileTx, Geinimi. In short, EC2 presents an early warning system for emerging new malware families, as well as a robust predictor of the family (when it is not new) to which a new malware sample belongs, and the design of novel strategies for data-driven understanding of malware behaviors.

Index Terms—Android, malware, ensemble, classification, clustering

1 INTRODUCTION

WITH over 1.4 billion active phones worldwide [1] and over 290,000 phones sold in Q1 2016 alone with an 84.1 percent market share [2], Android dominates the mobile market. But this success has a dark side—more than 97 percent of mobile malware target Android devices [3] in order to steal money or private data. Examples include illegally sending SMSs to premium-rate numbers, stealing calendars, emails, texts, contact lists, social network accounts, documents and banking credentials. To elude detection, malicious developers use *obfuscation* techniques (e.g., polymorphism and metamorphism) [19] to automatically generate multiple variants of the same malware, thus creating a new *family* [17] of malware samples having the same purpose but slightly different characteristics (e.g., different file hash, names of functions and variables). In fact, almost 9,000 new Android malware samples were found daily in 2016, an increase of 40 percent over 2015 [8].

Given the increasing number of new malware samples, manual investigation is impractical and might lead to significant delays in the release of detection signatures for anti-malware tools. The only viable way to manage such huge volumes of malware is to design novel algorithms

that can automatically group similar samples into families. This has several major benefits—(i) if a sample belongs to a known family, the same removal techniques can be re-used; (ii) security analysts can focus their manual investigation on the few new samples that do not belong to any known family, thus optimizing their limited time and resources; (iii) understanding the characteristics of each family helps to detect more robust signatures for anti-malware tools.

Most literature on mobile malware analysis [11], [14], [21], [48], [57] is focused on *detection* (i.e., distinguishing malware from benign software). Several efforts [28], [29], [30], [54], [55], [57], [58] have been proposed in the context of detecting Android malware families. Some of these works consider outdated datasets (e.g., [30], [58]) and/or do not characterize malware families through feature explanation (e.g., [28]), mostly because they use features that are low level and hard to interpret, such as API call sequences. However, the major limitation of all existing works [28], [29], [30], [54], [55], [57], [58] is that they focus only on large families (e.g., size > 10) for which training data is available, and ignore small families which represent the novel malware variants on which security analysts should focus their attention.

We propose EC2, the first algorithm that effectively classifies a malware sample into both large and small families (even if previously unseen). We begin by presenting a thorough performance comparison on the classification of Android malware families¹ by using DREBIN [4], [14], the

- T. Chakraborty is with the Department of Computer Science and Engineering, Indraprastha Institute of Information Technology, Delhi (IIIT-D), Delhi 110020, India. E-mail: tanmoy@iiitd.ac.in.
- F. Pierazzi is with the Department of Engineering “Enzo Ferrari”, University of Modena and Reggio Emilia, Modena 41125, Italy. E-mail: fabio.pierazzi@unimore.it.
- V.S. Subrahmanian is with the Department of Computer Science, Dartmouth College, Hanover, NH 03755. E-mail: vs@dartmouth.edu.

Manuscript received 29 Nov. 2016; revised 19 July 2017; accepted 2 Aug. 2017. Date of publication 21 Aug. 2017; date of current version 18 Mar. 2020. (Corresponding author: Tanmoy Chakraborty.)
Digital Object Identifier no. 10.1109/TDSC.2017.2739145

1. It is important to observe that in some cases the concept of “malware family” may be fuzzy, as a security analyst may consider that a sample belongs to more than one family. For this reason, all results in this paper refer to the publicly available and labeled DREBIN [4] and Koodous [9] datasets.

largest publicly available Android malware dataset with labeled families. Several state-of-the-art classification and clustering algorithms are evaluated for the task of family classification by considering different combinations of *static* and *dynamic* features. We then design EC2 as an ensemble that combines the best of classification and clustering, and evaluate its performance both on DREBIN [4] and on the more recent Koodous academic dataset [9], outperforming several comparative baselines [30], [42], [51], [59].

The paper makes five major contributions.

- (i) We thoroughly assess performance of several state-of-the-art supervised classification and unsupervised clustering algorithms for Android malware family identification. The best supervised classifier (Random Forest) has accuracy² of 0.93 for large families (size ≥ 10), which decreases to 0.73 for small families (size < 10) due to lack of training data. In contrast, the best unsupervised clustering method (DBSCAN) achieves accuracy of 0.91 for small families and 0.86 for large families (see Table 5).
- (ii) While past research has looked at clustering and classification separately for traditional (mostly PC) malware, EC2 combines the results of both clustering and classification to overcome the drawbacks of each. EC2 delivers significant performance across all malware families (including families with just one sample), achieving an overall accuracy of 0.97, and outperforming several comparative baselines on DREBIN.
- (iii) Unlike all related works [28], [29], [30], [54], [55], [57], [58], EC2 is the first algorithm that can classify malware samples into small or even *previously unseen* Android malware families. Given the huge number of malware released in the wild everyday [8], the capability of detecting small and previously unseen malware families is critical for prioritizing manual inspection of new threats. We show how we identify samples belonging to very small malware families—in fact showing 0.43 Precision and 0.31 Recall for detecting families of size just 1.
- (iv) We show how to characterize different malware families by extracting family-specific features that distinguish one family from others. We chose to use the DREBIN dataset as each sample is labeled with a ground truth family to which the sample belongs [4]; our analysis reveals that the most important features for malware family classification are: re-using signatures for signing malware variants, requesting network permissions, requesting permissions to read/send SMS messages (used to send texts to premium-rate numbers) and use of encryption (often used for string decryption or repacking of malicious code to avoid static analysis). We also show that some malware families are better identified through static features, while others require dynamic analysis (e.g., because they adopt string decryption at runtime of CnC server URLs and CnC commands to avoid trivial detection via static code analysis). We observe

that more recent malware families may show different behaviors trends [54] as obfuscation strategies become more sophisticated and adapt to anti-virus strategies; however, the approach we propose for family characterization is general and can also be applied to more recent malware samples.

- (v) To show that EC2 is equally efficient in detecting recently released malware samples, we use Koodous, a dataset (Dec 2015 to May 2016) of recent Android malware samples. Here too, EC2 outperforms other baselines with an overall F-Score of 0.74.

The rest of the paper is organized as follows. Section 2 presents a thorough literature review on malware detection and classification. Section 3 presents the DREBIN dataset. Section 4 describes static and dynamic features for malware samples. Section 5 shows how classification and clustering leads to our new EC2 algorithm. A detailed evaluation of the classification and clustering algorithms are separately presented in Sections 6 and 8 respectively. Section 7 shows how to use classification to automatically characterize malware families. Finally, Section 9 compares EC2 with several existing baselines, including the best classification and clustering algorithms and shows that EC2 outperforms all. Section 10 presents conclusions and directions for future works.

2 RELATED WORK

We identify and discuss three main areas of related work: (i) malware spread and characterization, (ii) malware analysis and detection, (iii) prediction of malware families.

Malware Spread and Characterization. Some broadly related works predict malware spread [40], [61], evaluate infection rates and risk indicators [56], characterize of malware on different third-party Android markets [45] etc. However, these works do not propose ways to detect/classify malware into into families.

Malware Analysis and Detection. Several efforts ([14], [15], [41], [44], [53], [60]) focus on *malware detection*, i.e., given a new sample, predict whether it is benign or malicious. Kolter et al. [41] propose an n-gram approach based on the bytes of malware binaries. Baldangombo et al. [15] propose an ensemble classifier relying on static features from Windows malware. Siddiqui et al. [53] compare some supervised classifiers using static features and achieve best performance with decision trees. Ye et al. [60] combine file content and file relations for malware detection. While these works focus on PC malware, malware detection for *mobile* devices [52] has gained much recent attention. Some works [22], [36], [46], [50] study the effectiveness of existing antivirus solutions against obfuscation techniques applied to malware, whereas our main focus in this paper is related to family prediction algorithms. Some papers [35], [49] proposed distillation and re-training on adversarially crafted samples to increase classifier robustness against obfuscation; but this approach has been later shown to be not effective [23] through a quantitative evaluation. Other works like DREBIN [14], Crowdroid [21], DroidAPIMiner [11] perform malware detection through static features of Android applications, whereas TaintDroid [32] and DroidScope [57] use dynamic features extracted from execution traces. DroidSIFT [62] performs semantics-aware classification based on control flows.

2. We report accuracy in terms of Macro F-Score (MaF*), formally defined in Section 8.

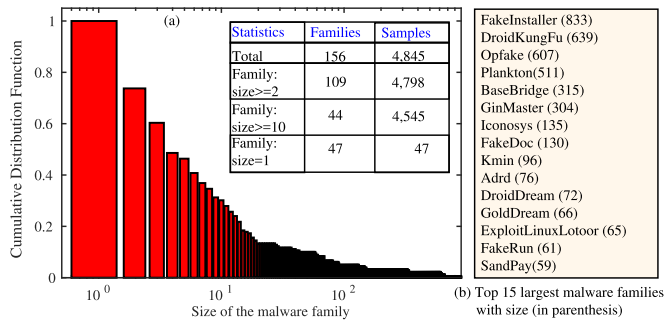


Fig. 1. (a) Cumulative distribution of the size of malware families present in the DREBIN dataset (see Supplementary, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TDSC.2017.2739145>, for the non-cumulative family distribution), (b) top-15 malware families by decreasing size.

MARVIN [44] proposes a binary classification method to discriminate between benign and malware applications using both static and dynamic features. MaMaDroid [48] builds behavioral models from dynamic logs through HMM to distinguish between benign and malicious applications. Unlike malware detection, our paper focuses on the problem of identifying malware families. Moreover unlike most past work we present a thorough analysis of several supervised classification and unsupervised clustering methods with different combinations of static and dynamic features.

Prediction of Malware Families. Several efforts ([17], [38], [51], [59]) consider the problem of *malware classification*, i.e., given a set of malware samples, identify which samples are variants of the same malware *family*. Ye et al. [59] propose an ensemble based on k -medoids and hierarchical clustering. Bayer et al. [17] propose a scalable method combining hierarchical clustering and locality-sensitive hashing on malware execution traces. Rieck et al. [51] propose a supervised SVM-based approach that clusters dynamic features from execution traces. All these works are focused on traditional PC malware, whereas our focus is on Android [52], an ecosystem that differs from traditional PC malware in terms of application, sandboxing, resource access, and system calls (more details in Section 4.1). Some recent work tackles the problem of Android malware classification. [16] proposes a very preliminary model checking approach to classify malware samples from just 2 families (OpFake, DroidKungFu) with 100 samples each. We consider 156 families including 47 singleton families with about 5,000 samples. Dendroid [55] proposes a text-mining based method to find similarities in malware families by extracting code chunks and deriving static features from them. DroidLegacy [30] proposes an approach to detect malicious code modules, but their focus is explicitly on piggybacked and repackaged applications. However, these works [30], [55] have some major shortcomings—they evaluate their approach only on the outdated MalGenome dataset [6], [63] (which is only a small subset of DREBIN, consisting of about 20 percent of the DREBIN dataset samples), and results obtained with their approach are hard to interpret for a human security analyst, whereas we automatically identify specific characteristics that distinguish malware families and which are easy to interpret. Moreover, past works do not consider small families, which is one of the major strengths of EC2. In [30] the authors remove all families with ≤ 10 samples, while in [55] the authors remove all

singletons (i.e., families with just one sample) from the analysis. There are other works related to mobile malware classification. DroidMiner [58] analyzes fine-grained behaviors of Android malware applications. DroidScribe [28] considers only dynamic features and classifies Android malware into families to provide a baseline against proposals that focus on static features. DroidSieve [54] considers only static features—in particular it proposes a set of features that are resilient to modern obfuscation strategies. Prescience [29] builds on DroidSieve and studies periodic retraining of classifiers over time to adapt to novel obfuscation techniques. However, in this paper we also consider small and previously unseen families which are not the main focus of [28], [29], [54], [58], [62]; in particular, our major contribution is the EC2 algorithm which is very efficient in detecting both very small (including new malware families) and large families. Andronio et al. [12] propose a method for discriminating between goodware, scareware and ransomware by generalizing three key insights of common behaviors in mobile ransomware: threatening text, device locking, and encryption. Aresu et al. [13] propose an approach that by extracting features from HTTP traffic is focused on classification of mobile botnets variants (e.g., Zitmo). However, these works are not generic since they are targeted to specific objectives (identify which malware is scareware/ransomware, and identify variants of mobile botnets samples), whereas we propose a generic approach that can be applied to any malware family and is also able to detect singleton families.

As a final remark, it is important to observe that in some cases the concept of “malware family” may be fuzzy, as a security analyst may consider that a sample belongs to more than one family. For example, consider the Petya ransomware and its more recent variant NotPetya [10], which antivirus vendors eventually decided to consider as two distinct families due to their differences in propagation and operativity. Nevertheless, the detection of small and/or previously unseen families can also be helpful in the detection of very different variants of a known family which may represent an entirely new strain. For the sake of fairness, all results in this paper refer to the publicly available and labeled DREBIN [4] and Koodous [9] academic datasets.

3 DREBIN DATASET

We first analyze the DREBIN dataset [4] as it represents the largest public, labeled mobile malware dataset, as of 2016. All DREBIN samples (even ones in the same family) are characterized by different *hash* values, indicating that some obfuscation technique [19] has been applied in order to prevent easy detection of variants. As described in Section 4.3.1, we execute the malware samples in a controlled environment to extract dynamic features. We only kept dynamic logs for samples executed for 120 seconds without failures. In particular, the final dataset used in this paper consists of 4,845 malware samples: Fig. 1a reports the distribution of malware family sizes, and Fig. 1b reports the top 15 malware families by size, among which: FakeInstaller, that simulates an installer of Android applications but in reality sends SMSs to a premium-rate service; DroidKungFu that tries to perform privilege escalation and steals sensitive data from

the device; Opfake that is another malware family sending SMSs to premium-rate numbers.

The size of Android malware families follows a heavy-tailed skewed distribution (cf. Fig. 1a): 112 of 156 families have under 10 samples. This suggests that supervised classification algorithms might not work well due to lack of training samples for small families. Hence, we leverage unsupervised clustering approaches and propose a novel ensemble method that leverages both classification and clustering.

4 FEATURE EXTRACTION

We first present some fundamentals about the Android environment, and then describe how we design and extract static and dynamic features from the samples.

4.1 Android Fundamentals

Android applications (or apk files) are *jar*-like compressed files. Android users can install applications from both official (*Google Play Store*) and third-party marketplaces. The *components* of each Android application must be declared in the `Manifest` file which is a header in each apk. The main apk components are:

- *Activities* model the screens and windows that constitute the user interface of an application.
- *Services* are background processes that remain active even when the application is out-of-focus (e.g., playing music), and may be used for malicious acts (e.g., aggressive advertisement from Adware).
- *Content Providers* are used to define and regulate data sharing between applications.
- *Broadcast Receivers* are used to monitor system-level events (such as `BOOT_COMPLETED`, commonly monitored by malware to trigger malicious behavior on phone startup [14]).
- *Intents* are messaging objects that can be used for interactions and communication between components. They are used to invoke *actions*, e.g., start another activity, start a service, or deliver a broadcast message that can be captured by broadcast receivers.

Each Android application runs in an *isolated* environment that is separated from other applications, and by default it can only write on a separate memory space and cannot access phone resources (e.g., Camera). To perform more advanced interactions, Android relies on a fine-grained *permissions system* for which an application has to explicitly request the user, at installation time, for permission to access specific *software* (e.g., access call log) and/or *hardware* (e.g., camera, vibration) resources. This guarantees transparency about the data that can be accessed by the application (e.g., access and modify Calendar or Contact List), although it is often overlooked by users since the number of permissions may be high. Moreover, it is not trivial to determine what an application actually does with a permission (e.g., when/where an app exactly uses `READ_SMS` permission to read text messages inbox). A full set of standard permissions is available in Android documentation,³ but

3. <https://developer.android.com/guide/topics/security/permissions.html>

program developers can also define a set of *custom permissions* depending on their needs (e.g., to communicate with other applications of the same developer).

4.2 Static Features

Several static features have been proposed in the literature (e.g., [14], [44], [54], [62]). We wanted to develop easy to extract and explainable static features by leveraging those proposed in [14]. Despite [14] extracts static features both from the Manifest and from the source code (e.g., API calls), we decided to consider only features from the Android Manifest⁴ for three main reasons: (i) features from the code may introduce overly detailed and noisy information [31], [36], whereas the Android Manifest already has rich information about an application and its structure; (ii) the use of encryption or reflection [46] may easily introduce much noise in the code, whereas the Android Manifest must be declared in plaintext and also contains many specifications about requested permissions and interfaces; (iii) finally, we consider features that are easy to interpret, so that we can automatically extract meaningful characteristics that distinguish malware families (cf. Section 7 and supplementary materials, available online).

Depending on their meaning, our static features fall into three main groups: *author*, *structure* and *permissions*. Table 1 reports the complete list of static features, where the left-most column reports feature groups.

Author. Malware of the same family may be developed by the same *author*. Developer information is usually published by marketplaces (e.g., Google Play). However, the DREBIN dataset also contains applications that have been removed from the store or that have been retrieved from third-party marketplaces. We assume that if two applications are digitally signed with the same certificate, then they have been developed by the same author. This feature has not been considered by [14], but in some other related research (e.g., [54]). Section 7 shows that some families (e.g., MobileTx and Opfake) have many samples developed by the same author—while the author feature is insignificant for others (e.g., Geinimi, FakeInstaller, JiFake) as the malware developers were more careful.

Application Components. Applications sharing similar structure may be variants of the same family. We describe app structure via the number of components found in the Manifest: `filesize`, `n_activities`, `n_intents`, `n_providers`, `n_receivers`, `n_services`. As an example, the feature `n_activities` is very significant to discriminate the family Opfake (cf. supplementary materials, available online), and also for discriminating the various families in the multi-class classification task (cf. Fig. 8).

4. Our static feature set corresponds to the features proposed in [14] related to the Android Manifest with some modifications: unlike [14], we also consider the `filesize` and `author` of an app (which have also been considered in [54]); unlike [14], we only consider the *number* of application components (e.g., Activities, Services), instead of their specific class *names* (e.g., `HomePageActivity.class`)—this is because names can be very easily obfuscated, whereas number of components may not be modified arbitrarily by the malware developer without raising suspiciousness in malware detectors. We also observe that all static features in [14] are binary, whereas we also have continuous static features: `author`, `filesize`, number of components, and number of permissions.

TABLE 1
Static Features Derived from the Analysis of Code
and Manifest in the Malware Samples in DREBIN

Group	Feature	Description
Author	author	Derived from SHA256 hash that digitally signs the Android sample
Structure	filesize	The size of the file in bytes
	n_activities	Num. Activities
	n_intents	Num. Filtered Intents
	n_providers	Num. Content Providers
	n_services	Num. Services
	n_receivers	Num. Broadcast Receivers
Permissions	n_std_sw_perm	Num. standard sw permissions required to install the application
	n_std_sw__perm_dangerous	Num. std sw permissions marked as dangerous in Android documentation
	n_hw_perm	Num. std hw permissions required to install the application
	n_custom_perm	Num. custom permissions (i.e., non-std) defined by the application developer
	sw_permission ₁	1 if sw_permission ₁ is required by the application, 0 otherwise
...	1 if sw_permission _i is required by the application, 0 otherwise	
sw_permission _N	1 if sw_permission _N is required by the application, 0 otherwise	
hw_permission ₁	1 if hw_permission ₁ is required by the application, 0 otherwise	
...	1 if hw_permission _j is required by the application, 0 otherwise	
hw_permission _K	1 if hw_permission _K is required by the application, 0 otherwise	

Unlike the feature set presented in [14], we choose not to consider the *names* of the Android components, but rather their counts because component names can be altered very easily (as they are just the names of Java classes defined by the application developer). On the other hand, the number and organization of components is harder to change significantly without also improving chances of detection of a sample as malware.

Permissions. The Android permissions system provides a rich source of features as malware from the same family may need the same permissions that may help distinguish them from other families. For example, `MobileTx` is one of the few families requiring the `RESTART_PACKAGES` permission, that is used to kill antivirus and application monitoring systems. Sometimes, even the absence of a certain permission requested by the malware can be useful to discriminate its family. For instance, `FakeInstaller` does not need the `ACCESS_NETWORK_STATE` permission—this absence is a strong indicator that a sample belongs to this family. A predefined set of *standard permissions*⁵ label some features *dangerous* as they provide access to sensitive resources such as call logs or contact lists. Our feature set includes binary vectors as in [14], containing possible standard software and hardware permissions from the Android official documentation. We also count the number of customized permissions that can be defined by programmers. As a final

5. <https://developer.android.com/guide/topics/security/permissions.html>

remark, although permission-related features are very important, they may not be sufficient by themselves for detecting/ classifying some types of Android malware (e.g., in case of privilege escalation attacks [20]); hence it is fundamental to consider the other kinds of features presented in this section as well.

We extract static features with official *Android SDK*⁶ tools and *Androguard*,⁷ a Python library for extracting metadata from apk files and their `Manifest`. We have 190 static features in all (Author: 1, Application components: 6, Permissions: 183).

4.3 Dynamic Features

Since static features alone may not be enough because of obfuscation techniques [19], we run malware samples in a sandbox in order to find common behaviors exhibited by families [51]. We first describe generation of dynamic logs, then design and motivate extraction of dynamic features.

4.3.1 Generation of Malware Dynamic Logs

We installed and configured the official Android emulator,⁸ a fully-functional Android system that runs applications with a graphical user interface. We also used `inetsim`,⁹ a tool that simulates many network services, and also provides realistic data in response to malware requests (e.g., if a malware tries to download an executable file from an external website). This allows us to log the most relevant malware Internet requests of the malware, despite absence of real Internet connectivity. To run the malware samples, we use `DroidBox 4.1.1`,¹⁰ an open-source sandbox especially tailored for dynamic malware analysis of Android applications. `DroidBox` can install and execute apk applications in the Android emulator. For each malware sample in DREBIN/ Koodous, we execute the malware for *120 seconds* and log all its activities through `DroidBox`, which include: file system activities (read/write), network activity (sendnet/recvnet), usage of cryptographic primitives, dynamic loading of classes, start of new services (background processes), generation of system events. We did not use simulated user input because: (i) a *random* input simulator would prevent deterministic code execution, which is important for malware classification; (ii) designing a *deterministic* input simulator to be run on all applications would be a huge challenge as many malware applications *crash* often.

Fig. 2a shows a sample `Droidbox` log of a filesystem *write* operation, where about 1.9s after starting the execution of the malware, some data is written in `/data/com.apsp/stats.log`. The “data” field in Fig. 2a contains the data written by the application in hexadecimal format. Fig. 2b shows another sample log related to SMS sending activity. Many malware try to deceptively send SMSs to premium-rate numbers owned by the malware developer in order to get money from the user. Sometimes, SMSs are also used to spread the malware by sending a text to users from a contact list to deceive them into clicking a malicious URL.

6. <https://developer.android.com/studio/>

7. <https://github.com/androguard>

8. <https://developer.android.com/studio/run/emulator.html>

9. <http://www.inetsim.org>

10. <https://github.com/pjlantz/droidbox>

<pre>fdaccess: { 1.9002759456634521: { data: 541545301efbfb7a2a57, id: 1590266696, operation: write, path: /data/com.app/stats.log, type: file write},... }</pre>	<pre>sendsms: { 12.9806270599365234: { message: "Thanks for downloading Xmas walls!", number: <phone-number>, type: sms }, ... }</pre>
(a)	(b)

Fig. 2. Example of (a) write-log and (b) SMS activity log generated by DroidBox.

Details of the dynamic log generation process can be found in the supplementary materials, available online.

4.3.2 Dynamic Feature Extraction

Our goal is to design features that capture similarities in malware behaviors in order to classify samples into families. Dynamic logs allow us to even design very low-level features, such as read operations on specific data from files and folders at particular timestamps. However, past work shows that overly detailed dynamic features rarely improve malware classification [28], [31] as they inject noise. We therefore leverage an n -gram representation commonly used in malware analysis and classification (e.g., [18], [31], [33], [51]). Using n -grams, we *count* the following operations captured by the sandbox: RW (read and write operations), System (e.g., start of a new background process), Network (Internet requests), SMS (texts sent by the device). For each dynamic log operation, we extract several possible sets of words to limit the impact of possible obfuscation strategies adopted by the attacker (e.g., changing the filename of a written file). For this purpose, we consider a *bag of word* approach in which we extract n -gram counts. In particular, we have experimentally verified that considering n -grams with $n > 3$ does not yield any performance improvement (because the level of details increases and features become too specific, as discussed in [28], [31]), so we consider unigrams, bigrams and trigrams. Each word is a feature, where the value is represented by the number of occurrences of

that keyword in the logs [51]. The idea is that similar malware execute similar types of operations.

To clarify how the bag of words for dynamic features are extracted, consider an example based on the write log of Fig. 2a. Since we consider up to 3-grams, we can extract the following four words from this log: (i) write, (ii) write /data/, (iii) write stats.log, (iv) write /data/stats.log. The first word corresponds just to the name of the executed operation. In the second word, we extract only the *basepath* /data/ instead of the *fullpath* /data/com.app/ (i.e., the first folder name in the path, in addition to the root folder) because most Android folders are dependent on Application names (e.g., com.app in the example) or process ids, but the basepaths are unique [31]. We do not include the content written - we verified that it is not useful for malware classification, as many malware read files in different sequences (e.g., a 256 byte file read 1 or 2 bytes at a time), and also use encryption strategies to hide the real read/written content. Note that feature values represent *number of occurrences* of a word in the dynamic log of the malware sample.

We can then define a generalized version of a write operation feature set as follows:

$$\text{write} \left[\left[\langle \text{basepath} \rangle \right] \left[\langle \text{filename} \rangle \right] \right], \quad (1)$$

where items between squared brackets are optional (hence, if one considers all the possible combinations she can obtain four words).

Table 2 reports the full list of dynamic feature types extracted with the bag of words approach. The rationale behind the design of these features is to capture similarities in malware behaviors by counting the number of high-level actions of each sample [31]. Section 7 shows that dynamic features are effective in characterizing different malware families (cf. supplementary materials, available online): for example, Opfake loads a malicious apk in memory at runtime through the dexclass loading function; MobileTx sends SMS to premium-rate numbers; Geinimi starts a

TABLE 2
Dynamic Features Derived from the Execution of the Android Malware Samples Using Bag of Words

Group	Feature	Description
RW	read [[<basepath>] [<filename>]]	N -gram counts about read operations on the Android filesystem. The <code>basepath</code> is just the name of the first folder after the root
	write [[<basepath>] [<filename>]]	N -gram counts about write operations on the Android filesystem. The <code>basepath</code> is just the name of the first folder after the root
System	servicestart [<servicename>]	N -gram counts about started background processes (i.e., services)
	load [[<classpath>] [<classname>]]	N -gram counts about dex Android classes loaded during execution
	crypto [[<algorithm>] [<encryptionkey>]]	N -gram counts about crypto operations performed during execution
	recvsaction [[<path>] [<name>]]	N -gram counts about event listeners (e.g., <code>BOOT_COMPLETED</code>)
Network	sendnet [[<protocol>] [<port>]]	N -gram counts about outgoing network activity
	recvnet [[<protocol>] [<port>]]	N -gram counts about incoming network activity
SMS	sendsms [[<phonenumber>] [<message>]]	N -gram counts about sms sent by the application

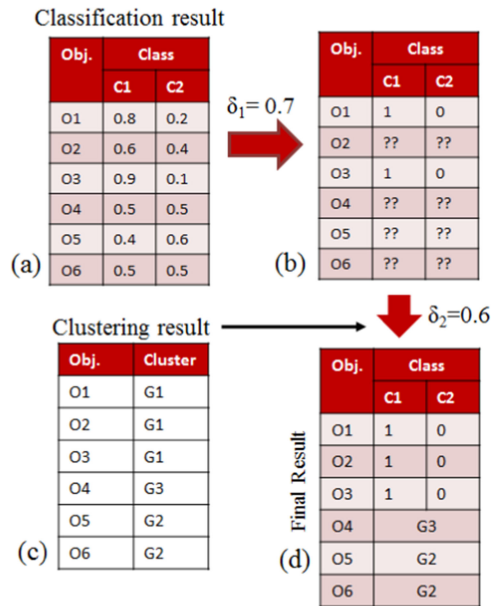


Fig. 3. An illustrative example EC2. Six samples O_1, \dots, O_6 are finally grouped into three clusters C_1, G_2, G_3 .

fake background process called GoogleKeyboard that collects user information that are then sent outside.

Note that since we always consider 1-gram counts consisting of just the operation (e.g., write, read), our approach also captures the total number of dynamic operations of each kind executed by a malware.

By executing DREBIN malware samples, we collect 6,875 dynamic features corresponding to the set of possible words of Table 2. We then drop all the features that have value 0 (i.e., never occurred) for all malware samples except one. In other words, we drop all n -grams executed by just a single malware sample in the dataset - as they do not contribute to the malware classification task; thus we are left with 2,048 dynamic features.

5 DISCOVERING ANDROID MALWARE FAMILIES

5.1 Supervised Classification

We consider six standard supervised classifiers—Decision Tree (DT), K-Nearest Neighbors (K-NN), Logistic Regression (LR), Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF). We perform hyperparameter optimization in order to find the parameters that generate the best results. For instance, we use CART with Gini gain criteria for DT; K-NN method with $K = 5$; multinomial logistic regression and SVM with linear kernel. Section 6 shows that Random Forest turns out to be the best classifier in general, but fails to capture small families. This motivated us to run unsupervised clustering algorithms with the hope that interdependency between malware samples captured via clustering might help in detecting small families.

5.2 Unsupervised Clustering

We consider five well known clustering methods previously used in malware analysis: [17], [26]: DBSCAN, Hierarchical with complete linkage and euclidean distance, Affinity, K-Means and MeanShift. The value of K in K-Means was

determined by the Silhouette Method. Other parameters were systematically tuned to get the best performance.

Section 8 shows that DBSCAN turns out to be the best clustering algorithm. Although it accurately captures small families, its overall performance is significantly worse than the best classification algorithm. This further motivates us to combine both the results of classification and clustering in a systematic way to improve the overall performance as well as to detect small families.

5.3 EC2: Combining Classification and Clustering

Section 6 will show that while classification methods perform well on families with at least 10 samples, they fail to predict small families effectively. On the other hand, clustering methods efficiently capture small families. To get the best of both worlds, we propose EC2 (Ensemble Clustering and Classification) which combines the results of classification and clustering in a systematic way (see Algorithm 1).

In the initial steps, EC2 uses an unsupervised clustering method A_{uc} to cluster all the samples (Step 1). In parallel, it trains a supervised classifier A_{sc} on training set TR (Step 2) and measures the membership probability of each test sample for each family (Step 4). If a test sample achieves a maximum membership probability greater than a certain threshold δ_1 , it is assigned to the corresponding family (Step 1); otherwise it is marked as “unlabeled” (Step 10). In Section 9.2, we vary the value of δ_1 and observe that the highest accuracy is obtained with $\delta_1 = 0.6$. For each such unlabeled sample s , EC2 first checks the cluster C_s where it is assigned by A_{uc} (Step 1). If more than δ_2 fraction of the constituent members in C_s have already been labeled with a single family f , then s is labeled with f (Step 1); otherwise s is labeled with a new family C_s . Note that in this step if there is a tie in the size of the classes inside the cluster C_s , we by randomly assign the sample into one of the majority classes.¹¹ Moreover, Step 1 enables us to create a completely new family which may not be present in the training set, thus allowing us to identify completely unobserved families.

Illustrative Example. Fig. 3 shows an example of EC2 in action. Suppose there are six samples O_1, \dots, O_6 . A classification algorithm A_{sc} classifies the samples with a probability distribution as shown in Fig. 3a. If $\delta_1 = 0.7$, then samples O_1 and O_3 are assigned to class C_1 because the membership probability of both these samples is above 0.7 (Fig. 3b). No other samples are assigned to a class due to the lack of high confidence. Therefore, we run a clustering algorithm A_{uc} that may group them into three clusters G_1, G_2, G_3 (Fig. 3c). For the unassigned samples we use the clustering results as follows. For instance, in the case of O_2 we see that it belongs to cluster G_1 , and there are two other samples O_1 and O_3 which also belong to G_1 . Now, out of 3 samples in G_1 , two are already labeled as C_1 from the classification result, i.e., more than δ_2 (which is set as 0.6) fraction of samples in G_1 are labeled as C_1 . Therefore, O_2 is also assigned to C_1 . However for O_5 and O_6 which belong to G_2 and there is no other member labeled earlier, we assign them separately in a class G_2 . Similarly O_4 is assigned to a singleton class G_3 .

11. Note that if δ_2 is greater than 0.5, there is no possibility that a tie is encountered.

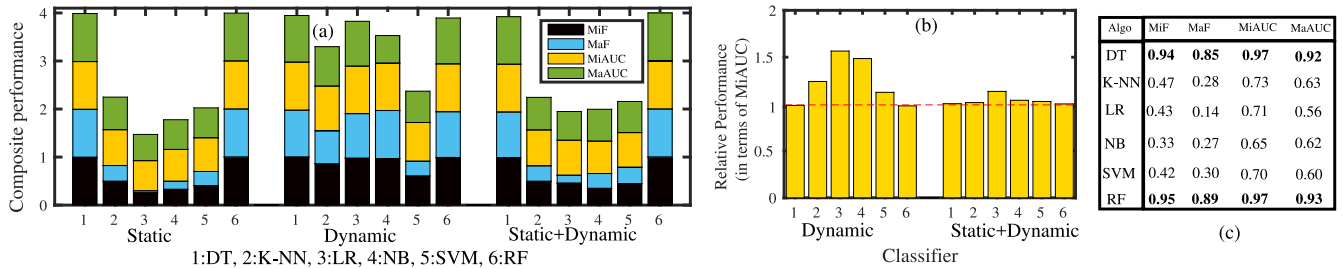


Fig. 4. Performance for families having ≥ 10 samples averaged over 50 iterations with five-fold cross-validation on the DREBIN dataset. (a) Composite performance of the classifiers; (b) relative improvement (in terms of Micro AUC) of the classifiers with respect to their performance with only static features (the horizontal dotted red line indicates that both the performance are equal); (c) absolute values of the metrics for all the classifiers considering static and dynamic features together.

Time Complexity. Assume that there are n malware samples and f families in the test set. Once we obtain the results of the classification and clustering algorithms, for each sample it takes $\mathcal{O}(f \log f)$ time to obtain the maximum membership probability. Therefore, the total time required to execute Steps 3-10 is $\mathcal{O}(nf \log f)$. Further if we assume there are on average n_c samples per cluster, the worst case total cost incurred by EC2 in Steps 11-17 is $\mathcal{O}(n.n_c) \sim \mathcal{O}(n^2)$. However, in practice $n_c \ll n$, which implies that $\mathcal{O}(n.n_c) \sim \mathcal{O}(n)$. So the overall worst-case time complexity is $\mathcal{O}(n + nf \log f) \sim \mathcal{O}(nf \log f)$.

Algorithm 1. EC2 Algorithm

Input: Training set: TR , test set: TS , thresholds: δ_1 and δ_2 ,
Classification algo: A_{sc} , Clustering algo: A_{uc}

Output: Labeled families of TS

- 1 Run A_{uc} on $TR + TS$ and obtain set of clusters \mathbb{C} ;
- 2 Train A_{sc} on TR ;
- 3 **for each** $s \in TS$ **do**
- 4 $MP[s, f] \leftarrow$ Membership probability of s in family f obtained from A_{sc} ;
- 5 $mp^* \leftarrow \max_{f} MP[s, f]$;
- 6 $f^* \leftarrow \underset{f}{\operatorname{argmax}} MP[s, f]$;
- 7 **if** $mp^* \geq \delta_1$ **then**
- 8 $L[s] \leftarrow f^*$; \triangleright Assigning family label of s
- 9 **else**
- 10 $Push(UnlabeledS, s)$ \triangleright Push s into $UnlabeledS$
- 11 **while** $UnlabeledS$ is not empty **do**
- 12 $s \leftarrow Pop(UnlabeledS)$;
- 13 $C_s \leftarrow$ Cluster of s (where $C_s \in \mathbb{C}$);
- 14 **if** More than δ_2 fraction of the samples in C_s are labeled with a family f **then**
- 15 $L[s] \leftarrow f$;
- 16 **else**
- 17 $L[s] \leftarrow C_s$;
- 18 **return** L ;

6 PERFORMANCE OF SUPERVISED CLASSIFIERS

In this section, we provide the performance of the classifiers. We start with the metrics used to evaluate the performance of the classifiers, followed by a comparative evaluation.

6.1 Evaluation Metrics

As the size distribution of malware families is highly skewed (cf. Fig. 1a), we examine the performance of the

classifiers at Macro (Ma) and Micro (Mi) levels. At each level, we consider Precision (P), Recall (R), F-Score (F) and Area under the ROC curve (AUC). At micro (*resp.* macro) level, Precision, Recall and F-Score are denoted as MiP, MiR and MiF (*resp.* MaP, MaR and MaF) respectively. Each of such metrics ranges from 0 (no match) to 1 (exactly similar).

We recall how Micro and Macro statistics are computed for evaluating multi-class classifiers. Given a classifier and a malware family i , we let TP_i , FP_i and FN_i indicate True Positive, False Positive and False Negative samples, respectively. Then, for n different families, Micro Precision (MiP) and Macro Precision (MaP) are

$$MiP = \frac{\sum_{i=1}^n TP_i}{\sum_{i=1}^n TP_i + FP_i} \quad (2)$$

$$MaP = \frac{\sum_{i=1}^n \frac{TP_i}{TP_i + FP_i}}{n} \quad (3)$$

We also observe that micro statistics suppress the results of small clusters over those of large clusters.

6.2 Performance Analysis

We observe that Random Forest (RF) achieves the highest overall classification accuracy after 5-fold cross-validation¹² (see Supplementary Materials, available online, for the performance of the other classifiers). RF yields a MaF=0.65 and MaAUC=0.83, and MiF=0.93 and MiAUC=0.97 averaged over 50 iterations for *all families* (including the ones with less than 10 samples). Small families lead to lower values for macro statistics because these are difficult to predict due to the lack of training data. Performance of small families will be separately discussed in Section 6.4. For now, we consider only families with size ≥ 10 (44 families, 4545 samples).

Composite Performance. Fig. 4c shows performance of all classifiers on families of size ≥ 10 considering both static and dynamic features. For better visualization we adopt the setup used in [25] - for each evaluation metric (such as MiF, MaF, MiAUC, MaAUC), we separately scale the scores of the methods so that the best performing method has a score of 1. The composite performance of a method is the sum of the four normalized scores. If a method outperforms all other methods, then its composite performance is 4

¹² We consider 5-fold over 10-fold because the former handles folds with less than 10 samples in our training data. It is required because in our dataset small families are the prevalent families.

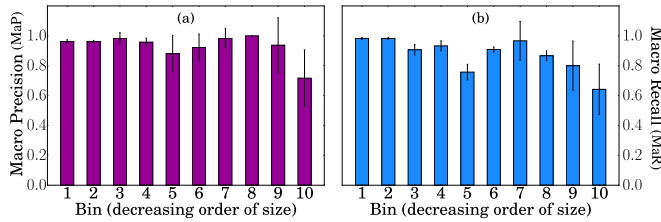


Fig. 5. Family-wise accuracy (mean values of (a) Macro Precision (MaP) and (b) Macro Recall (MaR) and their variance) of Random Forest (RF) with static and dynamic features on the DREBIN dataset. The families are grouped in bins and sorted by descending order of their size. The results are obtained with five-fold cross-validation and averaged over 50 iterations. The small drop in bin 5 is caused by `Boxer` and `SMSreg` families, that have variants harder to detect.

(See Supplementary Materials, available online, for the actual performance value of the classifiers).

Fig. 4a shows the composite performance of all the classifiers for different feature sets. Considering both static and dynamic features, Random Forest outperforms all others (with composite performance of 4), followed by DT (3.92), K-NN (2.24), SVM (2.15), NB (1.99) and LR (1.94). The superior performance of DT corroborates the results in [15], [41] for separating malware from benign applications and might be due to the categorical features such as author, structure and permissions. As described in [24], K-NN and DT are extremely useful when the features are categorical and/or a mixture of continuous and categorical. This might also explain the poor performance of SVM. Fig. 4b presents the relative performance of the classifiers by individually considering dynamic features and both static and dynamic features together w.r.t. the performance with only static features (which were used in most previous mobile malware classification research [30], [34], [55]).

Fig. 4b reports that for all classifiers the relative performance exceeds one in most cases, showing that dynamic features improve classification accuracy compared to static ones. Although the relative performance of K-NN, LR, NB and SVM is higher with only dynamic features, among all classifiers the best absolute performance is achieved by Random Forest with both static and dynamic features (Fig. 4c). Moreover, Section 7 will show that adopting both kinds of features is also extremely useful in distinguishing characteristics of each malware family. Therefore, unless otherwise mentioned, we will present results that include both static and dynamic features and that refer to the best classifier (RF).

Family-Wise Performance. To understand the detailed results of the classifier, Fig. 5 shows the family-wise accuracy of Random Forest, the best classifier. We consider all the detected (*resp.* ground-truth) families, sort them by descending order of size, and plot the Precision (*resp.* Recall) for the families with at least 10 samples in Fig. 5a (*resp.* Fig. 5b). We observe high Precision irrespective of family size, whereas Recall drops slightly for smaller families. The small performance drop at bin 5 for both Precision and Recall is caused by two malware families, `Boxer` and `SMSreg`, that have some hard-to-detect variants.

6.3 Prediction of Unknown Families

Thus far, out of total 4,845 samples we have considered 4,545 samples belonging to 44 large families (families with at least 10 samples). However, there are 300 other malware

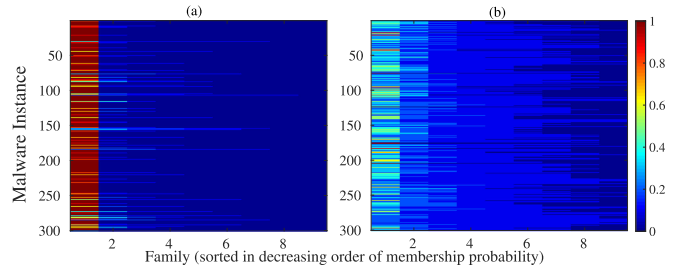


Fig. 6. Family membership probability of test samples (300 samples each from (a) known (Set II) and (b) unknown families (Set III)) obtained from Random Forest with static and dynamic features on the DREBIN dataset. For better visualization, we plot top 10 most probable families per test sample and sort families based on descending order of membership probability.

samples present in a total of 112 small families. In this experiment, we treat these 300 malware samples as “unknown malware samples”. In particular, out of 4,545 samples (with size greater than 10) we randomly choose 4,245 samples for our training set (set I). Two test suites are prepared for this experiment: (i) Set II: the remaining 300 samples from large families out of 4,545 sample set, (ii) Set III: 300 unknown malware samples (belonging to 112 small families) whose representatives are absent in the training set. The classifiers are trained on Set I to predict samples in Set II and Set III. The experiment is repeated 50 times to see how the classifier performs especially for Set III. We wish to show that a standard classifier should identify the family of a sample with *high confidence* if the malware belongs to one of the given families. However, if it does not belong to a known family, the classifier should assign it to a known family with *low confidence*.

We use Random Forest with all (static and dynamic) features to estimate the probability of each test sample belonging to different families. Fig. 6a shows that Random Forest with the proposed feature set is very confident in labeling Set II (the maximum probability is greater than 0.80 for the great majority of test samples; and MaF and MiF are 0.96 and 0.91); whereas in Fig. 6b it is extremely erratic in predicting the labels of Set III. As we have seen in Section 5.3, this behavior of the classifier has been leveraged to design the EC2 algorithm.

6.4 Performance for Small Families

Thus far, we have primarily considered 44 families with size ≥ 10 . However, there are 109 families that have size ≥ 2 , and each of the remaining 47 families has only one sample (singleton families).

We now consider all the families of size ≥ 2 and adopt a two-fold cross validation using stratified sampling so that at least one sample per family is present in the training set. The experiment is repeated 50 times and the average accuracy is reported. We separately measure class-wise performance. Fig. 7 shows the performance of Random Forest for families of size less than x ($5 \leq x \leq 50$) in order to see how the classifier performs on such heavily skewed families.

We compare this performance with the performance obtained earlier in Fig. 4 for families with at least 10 malware samples. Fig. 7 shows that the performance of the classifier drops significantly if we consider only small families in the test set—not surprising due to the small size of the

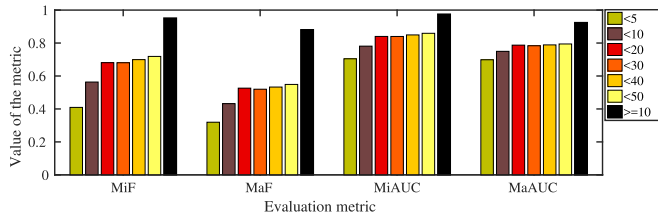


Fig. 7. Performance of Random Forest (RF) for families with different sizes, averaged over 50 iterations with two-fold cross-validation on the DREBIN dataset. We compare the performance of RF considering only small families (having size less than x , where $5 \leq x \leq 50$) with large families (having size ≥ 10).

training set. Moreover, although Section 6.3 shows that for samples belonging to unknown families Random Forest is erratic in assigning them into known families, it is not possible for a classifier to identify completely unknown families. This means that singleton families cannot be identified by classifiers. However, because of the huge daily growth in malware variants, it is critical to identify malware families as soon as an application has been detected as malware. This motivates us to study unsupervised clustering algorithms, as presented in Section 8.

7 CHARACTERIZING SOME MALWARE FAMILIES

A security analyst needs both an accurate predictor and to understand which features discriminate between different malware families. These findings are useful in engineering robust signatures for detection of new malware variants.

Fig. 8 reports the top 25 features by decreasing classification weight for the Random Forest multi-class classifier, with static features in *yellow* and dynamic features in *blue* (see Tables 1 and 2 for detailed features description). We see that static features are more important than dynamic ones, probably because Android applications have a rich set of information in their Manifest (i.e., XML-header), making it more challenging for malware authors to create hugely different variants. Fig. 8 shows that the most relevant static features are about *custom permissions*, that can be used for interactions and data sharing between applications with the same author/signature. *filesize* is also relevant, suggesting that variants of the same malware often share similar size distributions. Another important feature is *author*, because malware authors might reuse the same signature to publish many variants of their malicious applications on multiple markets. The most important *dynamic features* include the number of *write* operations on the filesystem and *read cpuinfo*, i.e., the number of read operations of CPU characteristics (to learn details of the device hardware to determine possible exploits). The receiver for *BOOT_COMPLETED* signal is also a top feature - it is often used by malware developers to determine when the device is turned on in order to trigger malicious behavior [14].

Malware Family Characterization. Apart from the overall discriminative features obtained from the multi-class classification, it is interesting to identify the *specific* features that characterize and distinguish *each* malware family from the rest. To this end, we design the following experimental setup - for each of the 44 large malware families, we learn the best *binary* classifier (Random Forest) to distinguish

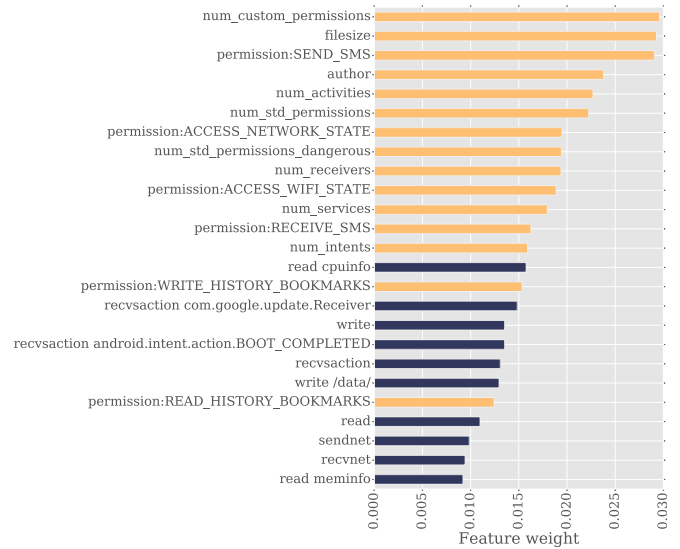


Fig. 8. Top 25 discriminative features of Random Forest multi-class classifier for the DREBIN dataset. Static and dynamic features are represented as *yellow* and *blue* bars, respectively.

each family from the rest, such that there exist 44 set of classification rules that separate the behavior of one malware family from all others. In this way, for each family we can sort the features by decreasing *weight* learned by the classifier, thus ranking the features that are more relevant for classification.

Fig. 9 shows relevant examples of the top 3 features for families of different sizes.¹³ Each row in Fig. 9 corresponds to a different family. Each histogram reports feature values on the *X*-axis, and the percent of malware samples having that feature value on the *Y*-axis. We report results about the following families¹⁴:

- *FakeInstaller* pretends to be an app that installs (or uninstalls) other apps from the system, whereas its primary purpose is to send premium-rate SMS without user consent.
- *MobileTx* is a Trojan that both steals information and also tries to send premium-rate SMS.
- *Geinimi* is a Trojan that opens a backdoor to send information from the device to a remote server.

Though some of these families share similar *gs* (e.g., sending premium-rate SMS), each of them is characterized differently through the top 3 features shown in Fig. 9.

FakeInstaller. (discovered in May 2012 [5]) The top 3 features in *FakeInstaller* are static (see first row of Fig. 9). The first feature is related to a permission accessing the network state (i.e., whether a connection is available or not on the device at a given time). This feature is important for discriminating *FakeInstaller* as it is one of the few malware families that does *not* require it (i.e., where permission *ACCESS_NETWORK_STATE* is set to 0, as can be observed in Fig. 9). The *filesize* of *FakeInstaller* samples is usually lower than the other families. Moreover, the *recvsaction*

13. The average F-score for the binary classification is even higher than multi-class classification, and is reported in supplementary materials, available online.

14. See supplementary materials, available online, for top 5 features and explanations of three other families.

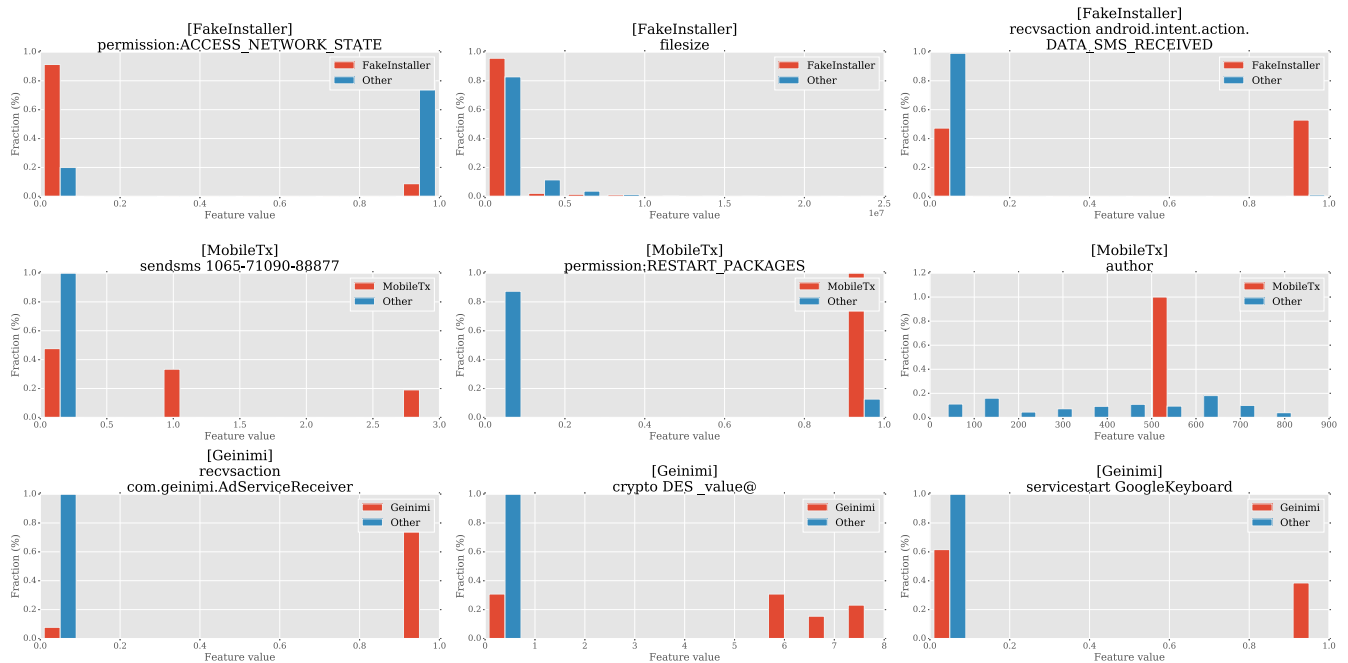


Fig. 9. Feature value comparison between malware families present in the DREBIN dataset. In particular, we report the histogram of the top-3 feature values for the following malware families (number of samples within parenthesis): FakeInstaller (883), MobileTx (21), Geinimi (13). (More families in the supplementary materials, available online). Each row corresponds to a family, and the feature value is compared against all other malware families in the dataset.

DATA_SMS_RECEIVED is used to monitor whenever an SMS is received, and is probably used by FakeInstaller to remove evidences of its texts sent to premium-rate number and to abort or delete incoming SMS.

MobileTx. (discovered in May 2012 [5]) The second row of Fig. 9 reports the top 3 distinguishing features of the MobileTx family, an SMS-fraud malware. The most important feature involves sendsms activity to the following phone number: 1065-71090-88877. This turns out to correspond to a premium-rate number used to steal money from the victim [37]. We discovered 68 premium numbers used by DREBIN samples. Some of these texts also have predetermined text content which is captured by our dynamic analysis. Moreover, all variants of MobileTx require the RESTART_PACKAGES permission that allows the malware to kill all processes including monitors and antivirus software installed in the device. The histogram in Fig. 9 shows that some other malware families (blue bar) also require this permission for the same purpose, but it is not common. Finally, all 21 variants of MobileTx have the same author=443 (that is the anonymized integer for the SHA256 signature), and hence this feature also helps identify this family.

Geinimi. (discovered in Dec 2010 [5]) Geinimi variants steal information such as fine location, device IDs (e.g., IMEI, IMSI) and the list of installed apps. Its top 3 distinguishing feature value distributions are shown in the last row of Fig. 9. The first feature is associated with the registration of AdServiceReceiver, a component used to monitor system events on which to trigger malicious behavior (e.g., to send stolen data to remove server). The crypto operations identified in the top 3 features use the DES algorithm to decrypt URLs of the CnC servers and GET commands which are encrypted to avoid static code analysis, but the malware authors recycled decryption keys and

hence we detected them during dynamic analysis. The permission MOUNT_UNMOUNT_FILESYSTEMS is used to access data in SD card slots, as some variants of Geinimi use SD cards to store and load repackaged malicious applications download from the CnC servers. The servicestart GoogleKeyboard represents the launch of a background process named GoogleKeyboard (to avoid user detection by looking at the phone task manager) that performs malicious actions, steals information periodically and sends it to the CnC whose URLs are decrypted at runtime.

Finally, some families are better distinguished using static features (e.g., FakeInstaller), whereas others are better distinguished through dynamic features (e.g., MobileTx). These statistics can be used by security analysts to define new signatures for malware classification as they automatically reveal some internals of the code obfuscation techniques adopted for a certain malware family.

8 PERFORMANCE OF CLUSTERING ALGORITHMS

In this section, we start by explaining the metrics used to evaluate clustering algorithms followed by a detailed performance evaluation.

8.1 Evaluation Metrics

We compare the detected clusters with the original families in terms of 5 well-known metrics: Micro F-Score (MiF*), Macro F-Score (MaF*), Normalized Mutual Information (NMI) [27], Adjusted Rand Index (ARI) [39] and Purity (PU) [47]. For MiF*, MaF*, NMI and PU (*resp.* ARI), the value ranges between 0 (*resp.* -1) and 1 where 0 (*resp.* -1) refers to no match with the ground-truth and 1 refers to a perfect match. Note, the definitions of Micro and Macro F-Scores for clustering are slightly different from the ones for the classification. They are defined as follows.

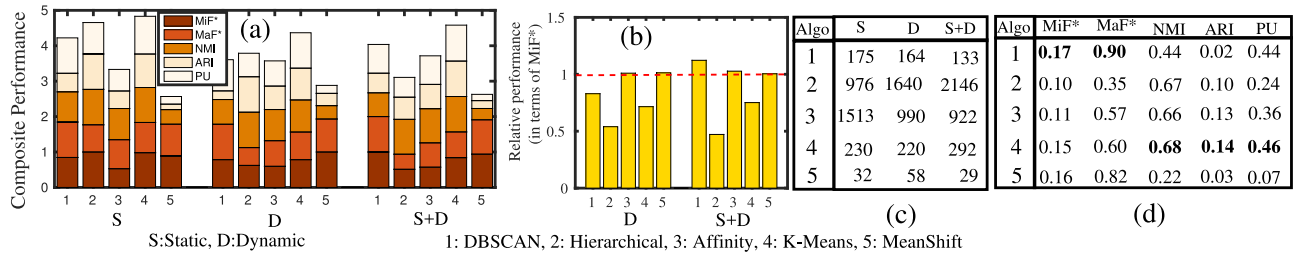


Fig. 10. (a) Composite performance of the clustering methods; (b) relative improvement of the clustering methods w.r.t. their performance with only static features; (c) number of clusters detected for different feature sets; (d) absolute value of the metrics for the clustering methods after considering both static and dynamic features together on the DREBIN dataset.

Given N samples and a detected cluster i with size n_i , True Positive (TP_i) denotes the number of pairs (out of $n_i C_2$) in which both samples belong to the same ground truth family; if not, they are counted as False Positive (FP_i). Similarly, True Negative (TN_i) counts the number of pairs that are outside cluster i , and that belong to different detected clusters j and k (i, j, k are mutually different), for which both samples belong to different ground-truth families. Once FP_i, TP_i, TN_i are computed for all detected clusters i , the calculation of MiF* and MaF* is the same as MiF and MaF (see Section 6.1).

8.2 Performance Analysis

We consider all malware samples in 156 families. Since clustering methods may produce different outputs depending upon the parameter settings and the initial seed selection, we run each clustering method 50 times, and the average performance is reported. Fig. 10 shows the composite performance (as discussed in Section 6.2) of all the clustering methods (see Supplementary Materials, available online, for the actual performance values). Among all, K-Means performs the best. Fig. 10b shows the relative performance of the 5 clustering methods using both static and dynamic features as compared to static features alone. Unlike classification, dynamic features alone do not improve performance. In 3 out of 5 cases, the combination of static and dynamic features leads to an improvement over using static features alone. This suggests that static features are more important than dynamic features in clustering malware. Fig. 10c shows the number of clusters generated by each method using both static and dynamic features. Fig. 10d summarizes the accuracy of different methods using both static and dynamic features. Overall, the performance of K-Means averaged over all the evaluation metrics is 0.41, followed by DBSCAN (0.39), Affinity clustering (0.37), Hierarchical (0.29) and MeanShift (0.26). This is in sharp contrast with

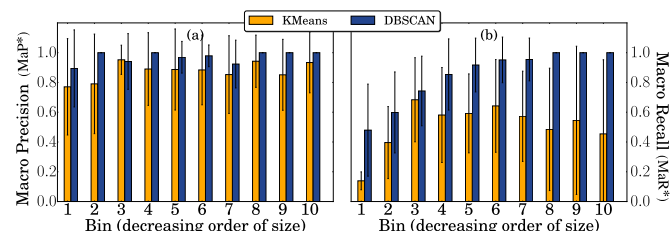


Fig. 11. Family-wise accuracy (mean values of (a) Precision and (b) Recall and their variance) of DBSCAN with static and dynamic features for the DREBIN dataset. The families are sorted in descending order of the size. The results are averaged over 50 iterations.

the results reported in [17] where hierarchical clustering yielded the best results for PC malware. A closer inspection of Fig. 10 reveals that DBSCAN dominates others in terms of Micro and Macro F-Score values. The reason might be that DBSCAN correctly identifies True Positives for each cluster/family which information theoretic metrics (such as NMI, ARI) often ignore as pointed in [43]. Moreover, the latter metrics do not penalize clusters with large cardinality and tend to prefer a small number of large clusters (which K-Means produces as shown in Fig. 12a later). Moreover, K-Means does not capture small families. We now discuss how the clustering methods detect small families.

Family-Wise Performance. Fig. 11a shows that irrespective of the size of the clusters detected, DBSCAN achieves better Precision than K-Means. Moreover, Fig. 11b shows that the DBSCAN's recall is almost double that of K-Means. These results emphasize the fact that DBSCAN accurately captures the skewed size distribution of Android malware families.

Performance for Small Families. Fig. 12a shows the cumulative distribution of the cluster size obtained from different methods as well as that obtained from the ground-truth families. DBSCAN produces the distribution which is closest to ground-truth in terms of Chi-square correlation (0.95). To identify which method best detects small families, we measure the performance for those clusters/families having size less than x (x varies from 10 to 50). Figs. 12b and 12c show that DBSCAN is the best. Interestingly, all clustering methods seem to be quite consistent in the small zones. These results highlight the necessity of adopting clustering methods for identifying small malware families.

Singleton Families. For singleton families, DBSCAN and K-Means seem to be the best and the worst methods

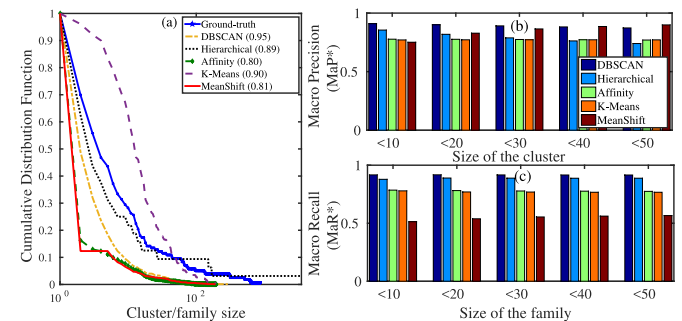


Fig. 12. (a) Cumulative distribution function (CDF) of the size of the clusters detected by different methods on the DREBIN dataset. The Chi-square correlation of the CDF obtained from each method with that of ground-truth is reported in the figure legend; (b) performance of the clustering methods for the small clusters/families having size less than x ($10 \leq x \leq 50$).

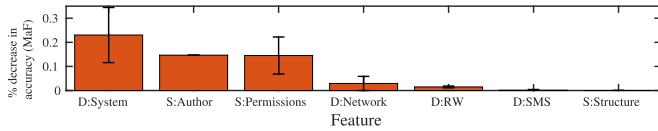


Fig. 13. Average performance (MaF*) decrease per group of static (S) and dynamic (D) features (DBSCAN) on the DREBIN dataset.

respectively. The Precision (Recall) of the clustering methods for detecting only singleton families are: DBSCAN: 0.33 (0.13), Hierarchical: 0.22 (0.07), Affinity: 0.06 (0.11), Mean-Shift: 0.02 (0.05) and K-Means: 0.002 (0.004). The detection of singleton families is very important and hugely challenging (due to lack of training data) because these identify potentially new types of malware that may need new anti-malware signatures.

8.3 Feature Importance

We identify the most important features in clustering by measuring the percentage decrease in accuracy (in terms of MaF*) when each feature is dropped. Fig. 13 reports results for DBSCAN, where the most important feature groups are D:System, S:Author and S:Permissions. Recall from Fig. 10b that combining static and dynamic features performs well, but dynamic features alone perform worse than static features. This differs from the classification results shown in Fig. 4b where dynamic features are extremely important. Thus both types of features are important.

To better understand the results in Fig. 13, let us look at the top 5 static and dynamic features in Table 3, along with their overall ranking (obtained by sorting for decreasing performance loss). Most of the top 5 dynamic features are related to dexclass operations, that correspond to the loading of particular Android classes during execution. Further inspection reveals that the first dynamic feature corresponds to a class loaded by 10 samples (out of 17) of the FakePlayer malware family, which is a Trojan that tries to send premium-rate SMSs. Table 3 suggests that very specific dynamic features (such as dexclass <class-name>) are used by DBSCAN to discriminate clusters. Therefore, samples with *very* similar features are grouped into *reasonably small* clusters, which are not allowed to grow further, resulting in high Precision and low Recall. The most relevant static features are related to Android permissions, such as those to access GPS location, Internet, send SMS or write to external storage. The author is fifth in terms of importance, meaning that it plays a significant role in unsupervised clustering approaches to grouping malware.

9 PERFORMANCE OF EC2

In this section, we present the performance of EC2. We start by explaining the baseline methods (Section 9.1), followed by a detailed comparative evaluation: Section 9.2 reports results on DREBIN dataset [4], Section 9.3 reports results on the more recent Koodous academic dataset [9].

9.1 Baseline Methods

We consider the following five methods as baselines: (i) best standalone classifier (Random Forest), (ii) RHWDL: an SVM based classification of malware behavior [51], (iii) HHC: an ensemble-based hybrid hierarchical clustering [59], (iv)

TABLE 3
Top 5 Features Each from Static and Dynamic Set with Their Overall Ranks (OR), Sorted by Decreasing Performance Loss of the DBSCAN Method for the DREBIN Dataset

OR	Static Feature	OR	Dynamic Feature
2	perm: ACCESS_COARSE_LOCATION	1	dexclass org.me.androidapplication1-1.apk
8	perm: INTERNET	3	dexclass ru.jabox.android.smsbox.lovebox-1.apk
10	perm: SEND_SMS	4	servicestart PlayerBindService
11	perm: READ_PHONE_STATE	5	dexclass ru.jabox.android.smsbox.jokebox-1.apk
12	author	6	dexclass com.agewap.soft-1.apk

DroidLegacy: an existing algorithm specifically designed for Android malware family classification [30], (v) K2: an ensemble approach where clustering results are used as features for classification [42]. While methods (ii), (iii) and (iv) were applied on malware datasets, method (v) was used for text classification. K2 turned out to be the best baseline.

9.2 Experimental Results on DREBIN

We consider the entire DREBIN dataset, randomly divide it into 5 folds and predict the families for each fold separately. The following experiment is repeated 50 times and results are averaged.

We assume that each test sample $s_i \in TS$ appears one by one and independently of other test samples. We always use the old and reliable training set TR to classify new test samples s_i , instead of augmenting with a newly classified sample.¹⁵ If s_i is classified into one of the existing classes, we keep s_i aside because other representatives of this class are already present in the training set TS . If s_i is classified into a completely new class, we use this information in a systematic way. When the next test sample s_{i+1} appears and is classified into a new class by the classifier (trained on TS), there are two possibilities - (i) either s_{i+1} belongs to the same class s_i belongs to, or (ii) s_{i+1} forms another new class. We then augment s_i and s_{i+1} into TS and run the clustering algorithm on $TS \cup \{s_i, s_{i+1}\}$. If both of them are clustered together, we assign them same class label; otherwise we keep them separately into two different classes. In this way, we avoid including noise due to old misclassification into the current training set.

We report the results for the best settings of the baseline methods—standalone Random Forest is the first baseline; we use the same configurations proposed in [51] and [59] for RHWDL and HHC respectively. For DroidLegacy [30] we extract the Android malware features, run their algorithm using their open-source implementation [7], and report best results obtained by tuning parameters as recommended in their paper. For K2 we report the results of Random Forest after considering outputs of *all* the clustering methods as features.

15. Augmenting a newly classified sample into the training set might lead to cascading of noise.

TABLE 4
Performance of the Baseline Methods and EC2
on the DREBIN Dataset

	Method	MiF*	MaF*	NMI	ARI	PU
Baselines	RF (Standalone)	0.63	0.84	0.63	0.13	0.41
	RHWDL [52]	0.53	0.81	0.60	0.09	0.41
	HHC [60]	0.68	0.79	0.59	0.08	0.39
	DroidLegacy [30]	0.67	0.82	0.62	0.09	0.35
	$\kappa 2$ [42]	0.72	0.89	0.63	0.11	0.42
EC2	RF+Hierarchical	0.71	0.94	0.69	0.14	0.47
	RF+Affinity	0.69	0.91	0.69	0.69	0.46
	RF+K-Means	0.67	0.90	0.70	0.14	0.47
	RF+MeanShift	0.73	0.93	0.68	0.14	0.46
	RF+DBSCAN	0.76	0.97	0.67	0.14	0.48

We consider Random Forest (RF) and different clustering methods separately with $\delta_1 = 0.6$ and $\delta_2 = 0.5$. The best baseline and the best combination for EC2 are highlighted.

Note that the number of detected clusters and the number of ground-truth families may not be identical (c.f. Fig. 3 in which there are two classes in the training sample; however we obtain 3 clusters in the final result). Therefore, we compare the performance of EC2 with other baselines using cluster evaluation metrics: MiF*, MaF*, NMI, ARI and PU (see Section 8 for formal definition).

Table 4 shows significant performance gains for EC2 (using Random Forest classifier and different clustering methods with $\delta_1 = 0.6$ and $\delta_2 = 0.5$) and other baselines. The values of the thresholds are selected based on Fig. 14—the performance of Random Forest is best for $\delta_1 = 0.6$ and $\delta_2 = 0.45$. The EC2 algorithm with DBSCAN turns out to be the best in this task—MiF*: 0.76, MaF*: 0.97, NMI: 0.67, ARI: 0.14 and PU: 0.48, which is 6, 9, 6, 27 and 14 percent higher than the best baseline ($\kappa 2$) respectively. Most importantly, EC2 detects singleton families with average Precision (Recall) of 0.43 (0.31), which is significantly higher than the performance reported in Section 8.2. Interestingly, DroidLegacy which was specifically proposed to classify Android malware families turns out to be even worse than $\kappa 2$ and EC2 when considering all the families. This indicates that ensemble approaches which combine clustering and classification results may be the best choice for classifying malware families.

Finally, we evaluated competing methods separately for small and large families. Table 5 shows that for families with size ≥ 10 , EC2 performs best, even better than Random Forest (standalone). For families with size < 10 , EC2 performs slightly worse than DBSCAN probably due to some classification noise propagated from the classifier. However, for small families EC2 still outperforms both RF and $\kappa 2$. Although we have shown the performance of EC2 for

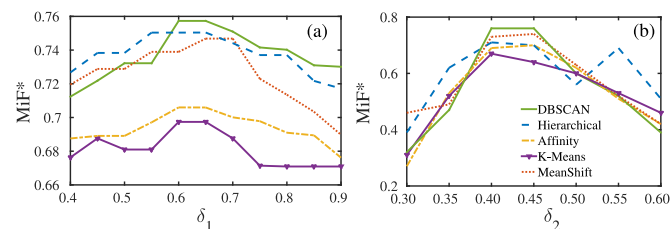


Fig. 14. Change in performance (MiF*) with the increase in (a) δ_1 and (b) δ_2 for EC2 (we consider Random Forest and different clustering methods) on the DREBIN dataset.

TABLE 5
Comparison of EC2 with the Best Classifier (Random Forest),
the Best Clustering (DBSCAN) and the Best Baseline Method
($\kappa 2$) for Small (Size < 10) and Large (Size ≥ 10) Families
in Terms of MiF* and MaF* on the DREBIN Dataset

Method	Small Families (< 10)		Large Families (≥ 10)	
	MiF*	MaF*	MiF*	MaF*
RF (Standalone)	0.29	0.73	0.78	0.93
DBSCAN (Standalone)	0.49	0.91	0.68	0.86
$\kappa 2$ [42]	0.36	0.75	0.76	0.90
EC2 (RF+DBSCAN)	0.45	0.89	0.82	0.95

Android malware prediction, we believe it can also be used to in general for other datasets.

9.3 Experimental Results on Koodous Dataset

We now present experimental results on the more recent Koodous dataset [9]. Koodous is a community-based platform similar to VirusTotal, but it is entirely focused on static and dynamic analysis of Android malware.

The Koodous academic dataset was generously provided to us by the Koodous administrators and contains 50,000 malware samples spanning from December 2015 to May 2016. This dataset is more recent than DREBIN, but only about 6,700 samples have been labeled with coarse-grained families (namely, *categories*). We use the labels already provided in this dataset as the ground truth because they have been closely investigated by Koodous administrators. In particular, the dataset contains the following categories (size within the parenthesis): SMS-fraud (3,257), Adware (3,069), Information-Theft (110), Ransomware (95), Fake-Installer (95), Rooting (50), Banker (31).

We use the Koodous dataset because (i) it contains more recent malware (hence it may be interesting to observe how EC2's performance changes as new obfuscation techniques are encountered), (ii) it contains more coarse-grained families which are more challenging to cluster.

We perform the same feature extraction techniques described in Section 4, and obtain static and dynamic features for the malware samples. We then consider 6,700

TABLE 6
Performance of the Baseline Methods and EC2
on the Koodous Dataset

	Method	MiF*	MaF*	NMI	ARI	PU
Baselines	RF (Standalone)	0.46	0.68	0.59	0.09	0.34
	RHWDL [52]	0.42	0.53	0.50	0.06	0.33
	HHC [60]	0.44	0.65	0.60	0.08	0.37
	DroidLegacy [30]	0.40	0.56	0.53	0.07	0.35
	$\kappa 2$ [42]	0.48	0.70	0.61	0.11	0.39
EC2	RF+Hierarchical	0.52	0.71	0.63	0.11	0.39
	RF+Affinity	0.51	0.69	0.62	0.10	0.38
	RF+K-Means	0.54	0.73	0.64	0.11	0.42
	RF+MeanShift	0.53	0.72	0.63	0.13	0.40
	RF+DBSCAN	0.56	0.74	0.65	0.13	0.45

We consider Random Forest (RF) and different clustering methods separately with $\delta_1 = 0.85$ and $\delta_2 = 0.5$. The best baseline and the best combination for EC2 are highlighted.

labeled malware samples, randomly divide it into 5 folds and predict the families for each fold separately. The following experiment is repeated 50 times and results are averaged.

Table 6 shows that although the overall prediction performance is lower than that of DREBIN, EC2 still outperforms the comparative baselines. Therefore, we may conclude that EC2 is useful for the datasets containing more recent malware samples (which are intelligently obfuscated and harder to detect).

10 CONCLUSIONS

This paper makes several major contributions. First, we proposed a *systematic analysis* of state-of-the-art classification and clustering algorithms applied to the task of grouping Android malware into families, and by considering different combinations of static and dynamic features. Second, we proposed EC2, the first algorithm used to group malware into families that uses *an ensemble of both classification and clustering approaches*. Third, we show that the performance of EC2 is high even when considering families of all sizes, achieving an overall Micro and Macro F-Score of 0.76 and 0.97, respectively. Moreover, EC2 also *works well on small malware families* which can be very hard to identify, yielding to an F-Score of 0.36 for singleton families which is very good, given the lack of training data. Fourth, a detailed analysis of some Android malware families reveals that our approach is also effective in characterizing and explaining behavior of Android malware families. Fifth, we showed that EC2 is equally successful in detecting families of more recent malware samples, and also performs significantly well with obfuscated feature set. Future work might involve how to group malware families into more coarse-grained categories and improve data-driven understanding of malware behaviors. Moreover, since malware authors may employ increasingly sophisticated obfuscation techniques, future work may also address how to design automated feature combination strategies to make it harder for the attacker to evade classification. For instance, instead of using a base set of features, we could replace them for analysis purposes with various linear or non-linear combinations of features.

ACKNOWLEDGMENTS

The authors would like to acknowledge the suggestions of the anonymized reviewers. Part of this work was funded by ONR Grants N000141612739, N000141512007, N000141612896, and N000141512742, ARO grant W911NF1410358 and Maryland Procurement Office grant H9823014C0137.

REFERENCES

- Oct. 2016. [Online]. Available: <https://techcrunch.com/2015/09/29/android-now-has-1-4bn-30-day-active-devices-globally/>
- Oct. 2016. [Online]. Available: <http://www.gartner.com/newsroom/id/3323017>
- Apr. 2017. [Online]. Available: <http://tinyurl.com/zqm6ufu>
- (2017, Jul.). [Online]. Available: <https://www.sec.cs.tu-bs.de/danarp/drebin/>
- Oct. 2016. [Online]. Available: <https://www.virustotal.com/>
- Android Genome Project (MalGenome). [Internet]. [Online]. Available: <http://www.malgenomeproject.org/>, Visited in Jul. 2017.
- DroidLegacy open-source implementation [Internet]. [Online]. Available: <https://bitbucket.org/srl/droidlegacy/src>, Visited in Jul. 2017.
- GDATA Mobile Malware Report, [Online]. Available: <https://www.gdatasoftware.com/news/2017/02/threat-situation-for-mobile-devices-worsens>, Visited in Jul. 2017.
- Koodous, Collaborative platform for Android malware analysts [Internet]. [Online]. Available: <https://koodous.com/>, Visited in Jul. 2017.
- Petya ransomware [Internet]. [Online]. Available: <https://www.nytimes.com/reuters/2017/07/05/business/05reuters-cyber-attack-ukraine-backdoor.html>, Visited in Jul. 2017.
- Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-level features for robust malware detection in Android," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2013, pp. 86–103.
- N. Andronio, S. Zanero, and F. Maggi, "HelDroid: Dissecting and detecting mobile ransomware," in *Proc. Int. Workshop Recent Advances Intrusion Detection*, 2015, pp. 382–404.
- M. Aresu, D. Ariu, M. Ahmadi, D. Maiorca, and G. Giacinto, "Clustering Android malware families by HTTP traffic," in *Proc. 10th Int. Conf. Malicious Unwanted Softw.*, 2015, pp. 128–135.
- D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and explainable detection of Android malware in your pocket," presented at the Netw. Distrib. Syst. Secur. Symp., San Diego, CA, USA, 2014.
- U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A static malware detection system using data mining methods," *arXiv preprint arXiv:1308.2831*, 2013.
- P. Battista, F. Mercaldo, V. Nardone, A. Santone, and C. Visaggio, "Identification of Android malware families with model checking," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 542–547.
- U. Bayer, P. M. Comparetti, C. Hlauschek, C. Krügel, and E. Kirda, "Scalable, behavior-based malware clustering," presented at the Netw. Distrib. Syst. Secur. Symp., San Diego, CA, USA, 2009.
- B. Biggio et al., "Poisoning behavioral malware clustering," in *Proc. Workshop Artif. Intell. Secur. Workshop*, 2014, pp. 27–36.
- J.-M. Borello and L. Mé, "Code obfuscation techniques for metamorphic viruses," *J. Comput. Virology*, vol. 4, no. 3, pp. 211–220, 2008.
- S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, and A.-R. Sadeghi, "XManDroid: A new android evolution to mitigate privilege escalation attacks," Technische Universität Darmstadt, Darmstadt, Germany, Tech. Rep. TR-2011-04, 2011.
- I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for Android," in *Proc. 1st ACM Workshop Secur. Privacy Smartphones Mobile Devices*, 2011, pp. 15–26.
- Z. Cai and R. H. Yap, "Inferring the detection logic and evaluating the effectiveness of Android anti-virus apps," in *Proc. 6th ACM Conf. Data Appl. Secur. Privacy*, 2016, pp. 172–182.
- N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.
- R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 161–168.
- T. Chakraborty, S. Kumar, N. Ganguly, A. Mukherjee, and S. Bhowmick, "GenPerm: A unified method for detecting non-overlapping and overlapping communities," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2101–2114, Aug. 2016.
- M. Chandramohan, H. B. K. Tan, and L. K. Shar, "Scalable malware clustering through coarse-grained behavior modeling," in *Proc. ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2012, pp. 161–168.
- L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Statist. Mechanics: Theory Experiment*, vol. 2005, no. 9, 2005, Art. no. P09008.
- S. K. Dash, et al., "DroidScribe: Classifying Android malware based on runtime behavior," in *Proc. IEEE Secur. Privacy Workshops*, May 2016, pp. 252–261.
- A. Deo, S. K. Dash, G. Suarez-Tangil, V. Vovk, and L. Cavallaro, "Prescience: Probabilistic guidance on the retraining conundrum for malware detection," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2016, pp. 71–82.
- L. Deshotels, V. Notani, and A. Lakhota, "DroiLegacy: Automated familial classification of Android malware," in *Proc. ACM SIGPLAN Program Protection Reverse Eng. Workshop*, 2014, Art. no. 3.
- M. Dimjašević, S. Atzeni, I. Ugrina, and Z. Rakamaric, "Evaluation of Android malware detection based on system calls," in *Proc. ACM Int. Workshop Secur. Privacy Analytics*, 2016, pp. 1–8.
- W. Enck, et al., "TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Trans. Comput. Syst.*, vol. 32, no. 2, pp. 1–29, 2014.

- [33] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in *Proc. IEEE Symp. Secur. Privacy*, 1996, pp. 120–128.
- [34] J. Garcia, M. Hammad, B. Pedrood, A. Bagheri-Khaligh, and S. Malek, "Obfuscation-resilient, efficient, and accurate detection and family identification of Android malware," Dept. Comput. Sci., George Mason University, Fairfax, VA, USA, Tech. Rep. GMU-CS-TR-2015-10, 2015.
- [35] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv:1606.04435*, 2016.
- [36] J. Hoffmann, T. Ryttilahti, D. Maiorca, M. Winandy, G. Giacinto, and T. Holz, "Evaluating analysis tools for Android apps: Status quo and robustness against obfuscation," in *Proc. 6th ACM Conf. Data Appl. Secur. Privacy*, 2016, pp. 139–141.
- [37] J. Hoffmann, M. Ussath, T. Holz, and M. Spreitzenbarth, "Slicing droids: Program slicing for smali code," in *Proc. 28th Annu. ACM Symp. Appl. Comput.*, 2013, pp. 1844–1851.
- [38] X. Hu, K. G. Shin, S. Bhatkar, and K. Griffin, "MutantX-S: Scalable malware clustering based on static features," in *Proc. USENIX Annu. Tech. Conf.*, 2013, pp. 187–198.
- [39] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.
- [40] C. Kang, N. Park, B. A. Prakash, E. Serra, and V. S. Subrahmanian, "Ensemble models for data-driven prediction of malware infections," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, 2016, pp. 583–592.
- [41] J. Z. Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *J. Mach. Learn. Res.*, vol. 6, pp. 2721–2744, 2006.
- [42] A. Kyriakopoulou and T. Kalamboukis, "Using clustering to enhance text classification," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2007, pp. 805–806.
- [43] V. Labatut, "Generalised measures for the evaluation of community detection methods," *Int. J. Social Netw. Mining*, vol. 2, no. 1, pp. 44–63, 2015.
- [44] M. Lindorfer, M. Neugschwandtner, and C. Platzer, "MARVIN: Efficient and comprehensive mobile app classification through static and dynamic analysis," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, 2015, pp. 422–433.
- [45] M. Lindorfer, et al., "AndRadar: Fast discovery of android applications in alternative markets," in *Proc. Int. Conf. Detection Intrusions Malware Vulnerability Assessment*, 2014, pp. 51–71.
- [46] D. Maiorca, D. Ariu, I. Corona, M. Aresu, and G. Giacinto, "Stealth attacks: An extended insight into the obfuscation effects on Android malware," *Comput. Secur.*, vol. 51, pp. 16–31, 2015.
- [47] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- [48] E. Mariconti, L. Onwuzurike, P. Andriotis, E. D. Cristofaro, G. J. Ross, and G. Stringhini, "MaMaDroid: Detecting Android malware by building Markov chains of behavioral models," in *NDSS*, San Diego, USA, pp. 1–15, 2017.
- [49] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *Proc. IEEE Military Commun. Conf.*, 2016, pp. 49–54.
- [50] V. Rastogi, Y. Chen, and X. Jiang, "DroidChameleon: Evaluating Android anti-malware against transformation attacks," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 329–334.
- [51] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *Proc. Int. Conf. Detection Intrusions Malware Vulnerability Assessment*, 2008, pp. 108–125.
- [52] A. Sadeghi, H. Bagheri, J. Garcia, and S. Malek, "A taxonomy and qualitative comparison of program analysis techniques for security assessment of Android software," *IEEE Trans. Softw. Eng.*, vol. 43, no. 6, pp. 492–530, Jun. 2017.
- [53] M. Siddiqui, M. C. Wang, and J. Lee, "Data mining methods for malware detection using instruction sequences," in *Proc. 26th IASTED Int. Conf. Artif. Intell.*, 2008, pp. 358–363.
- [54] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "DroidSieve: Fast and accurate classification of obfuscated android malware," in *CODASPY*, Scottsdale, AZ, USA, pp. 1–12, Mar. 2017.
- [55] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and J. Blasco, "Dendroid: A text mining approach to analyzing and classifying code structures in android malware families," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1104–1117, 2014.
- [56] H. T. T. Truong, et al., "The company you keep: Mobile malware infection rates and inexpensive risk indicators," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 39–50.
- [57] L. K. Yan and H. Yin, "DroidScope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic android malware analysis," in *Proc. USENIX Secur. Symp.*, 2012, pp. 569–584.
- [58] C. Yang, Z. Xu, G. Gu, V. Yegneswaran, and P. Porras, "DroidMiner: Automated mining and characterization of fine-grained malicious behaviors in android applications," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2014, pp. 163–182.
- [59] Y. Ye, T. Li, Y. Chen, and Q. Jiang, "Automatic malware categorization using cluster ensemble," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 95–104.
- [60] Y. Ye, et al., "Combining file content and file relations for cloud based malware detection," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 222–230.
- [61] S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojmenovic, "Malware propagation in large-scale networks," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 170–179, Jan. 2015.
- [62] M. Zhang, Y. Duan, H. Yin, and Z. Zhao, "Semantics-aware Android malware classification using weighted contextual API dependency graphs," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1105–1116.
- [63] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proc. IEEE Symp. Secur. Privacy*, 2012, pp. 95–109.



Tanmoy Chakraborty received the PhD degree from IIT Kharagpur, India, in 2015 as a Google India PhD fellow. He is an assistant professor and a Ramanujan fellow in the Department of Computer Science & Engineering, Indraprastha Institute of Information Technology Delhi (IIIT-D). Prior to this, he was a postdoctoral researcher with the University of Maryland, College Park. His PhD thesis was recognized as best thesis by IBM Research India, Xerox research India and Indian National Academy of Engineering (INAE). His broad research interests include data mining, social media, and data-driven cybersecurity. Home page: <http://faculty.iiitd.ac.in/tanmoy/>



Fabio Pierazzi received the master's degree in computer engineering and the PhD degree in computer science from the University of Modena and Reggio Emilia, Modena, Italy, in 2013 and 2017, respectively. He is a postdoctoral researcher with the University of Modena and Reggio Emilia, Modena, Italy. During 2016, he spent 10 months as a visiting research scholar in the Department of Computer Science, University of Maryland, College Park, Maryland. His research interests focus on security analytics, network security, malware classification, intrusion detection, and all solutions that combine cybersecurity and data mining. Home page: <http://weblab.ing.unimo.it/people/fpierazzi>



V. S. Subrahmanian is the Dartmouth College distinguished professor in cybersecurity, technology, and society. From 1989 to 2017, he was a professor in the Department of Computer Science, University of Maryland, College Park, having previously served as director of the University of Maryland Institute for Advanced Computer Studies (UMIACS). He was with the editorial boards of numerous ACM and IEEE journals as well as the *Science*, the board of directors of the Development Gateway Foundation (set up by the World Bank), SentiMetrix, Inc., and with the research advisory board of Tata Consultancy Services. He has worked extensively at the intersection of databases and artificial intelligence. He has authored more than 200 refereed publications, and is a fellow of the AAAI and the AAAS. Home page: <https://www.cs.umd.edu/users/vs/>

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Exploiting Dissent: Towards Fuzzing-Based Differential Black-Box Testing of TLS Implementations

Andreas Walz  and Axel Sikora 

Abstract—The Transport Layer Security (TLS) protocol is one of the most widely used security protocols on the internet. Yet do implementations of TLS keep on suffering from bugs and security vulnerabilities. In large part is this due to the protocol's complexity which makes implementing and testing TLS notoriously difficult. In this paper, we present our work on using differential testing as effective means to detect issues in black-box implementations of the TLS handshake protocol. We introduce a novel fuzzing algorithm for generating large and diverse corpuses of *mostly-valid* TLS handshake messages. Stimulating TLS servers when expecting a ClientHello message, we find messages generated with our algorithm to induce more response discrepancies and to achieve a higher code coverage than those generated with American Fuzzy Lop, *TLS-Attacker*, or *NEZHA*. In particular, we apply our approach to *OpenSSL*, *BoringSSL*, *WolfSSL*, *mbedTLS*, and *MatrixSSL*, and find several real implementation bugs; among them a serious vulnerability in *MatrixSSL* 3.8.4. Besides do our findings point to imprecision in the TLS specification. We see our approach as presented in this paper as the first step towards fully interactive differential testing of black-box TLS protocol implementations. Our software tools are publicly available as open source projects.

Index Terms—TLS, network security, cryptographic protocols, fuzzing, differential testing

1 INTRODUCTION

IN an increasing number of domains secure network communication is a key requirement, not only for functional safety but also for user acceptance. This is particularly true in the *Internet of Things* (IoT), where *Cyber Physical Systems* connect the physical world with the worldwide cyber space.

The *Internet Engineering Task Force* has developed the *Transport Layer Security* (TLS) protocol to provide *end-to-end* security over insecure networks [1], [2]. Since quite some time TLS is one of the Internet's cornerstones for secure communication. TLS supports encrypted and integrity-checked data transfer as well as mutual authentication of communication partners.

The TLS protocol is receiving continual scrutiny [3], [4], [5], [6], [7] and over time a long list of serious attacks on the protocol and some of its cryptographic constructions have been found [8], [9]. Still, in suitable and up-to-date configurations, TLS as a protocol can be considered sound and secure [10].

However, security vulnerabilities do not only arise from the protocol itself. Often, it is *the implementation* of the protocol which introduces serious security issues [11], [12]. This fact is emphatically demonstrated by another long list of

security vulnerabilities, this time caused by flaws in popular implementations of TLS, e.g., the *Heartbleed bug* [13] and the *CCS injection bug* [14] in *OpenSSL* [15], the *goto fail bug* [16] in *GnuTLS* [17], and others.

There are several reasons that make TLS implementations particularly susceptible to bugs:

- *Flexibility and agility*: The TLS protocol has a large parameter space with dynamic negotiation mechanisms. Several extension points allow adding new cryptographic algorithms and protocol features without touching the core specification.
- *Complexity*: The protocol's flexibility brings a high complexity to the protocol state machine, the handling of interactions between different protocol modes, and the message parsing. For example, TLS requires context-sensitive message parsers which are known to be a common source of bugs [18]. The parsing routines of TLS implementations can easily exceed several thousand lines of code.
- *Specification*: The specification of TLS along with all its extensions and options is distributed over nearly one hundred documents with thousands of pages. Beyond that does the specification not have formal character, lacking precision and leaving interpretation at the developer's discretion.
- *Interoperability and diversity*: Intolerant behavior in the diverse corpus of internet-deployed TLS implementations forces developers to individually trade strict conformance to the specification against large-scale interoperability.

• The authors are with the Institute of Reliable Embedded Systems and Communication Electronics (ivESK), Offenburg University of Applied Sciences, Offenburg 77652, Germany.
E-mail: {andreas.walz, axel.sikora}@hs-offenburg.de.

Manuscript received 5 Jan. 2017; revised 5 Oct. 2017; accepted 6 Oct. 2017.
Date of publication 17 Oct. 2017; date of current version 18 Mar. 2020.
(Corresponding author: Andreas Walz.)
Digital Object Identifier no. 10.1109/TDSC.2017.2763947

Unfortunately does testing TLS implementations turn out to be highly nontrivial; at least if more than mere *mostly-positive* testing is aimed for [19]. Some of the reasons are:

- *Protocol messages*: The complexity and peculiarity of the TLS message format demand sophisticated test message generators. TLS features a multitude of different message, content, and data types. Ignorantly generated test messages are likely to fail early input validation.
- *State machine*: The TLS protocol state machine requires dynamic and nontrivial protocol interactions for deep and thorough probing.
- *Cryptography*: By its nature, TLS makes extensive use of cryptography. The use of cryptography requires elaborate and dynamic test interactions, just like a complex state machine does.
- *Lack of formal reference*: Given the lack of a formal TLS specification, no entity exists that can reliably tell correct from incorrect implementation behavior. After all, who can reliably decide whether the outcome of a test is positive or negative?

In particular the last-mentioned issue is addressed by various research activities: for example, *miTLS* is a reference implementation of TLS verified with respect to certain security properties [20], [21]. However, it is unlikely that verified implementations like *miTLS* fully replace prevalent implementations like *OpenSSL*, *BoringSSL*, or *WolfSSL* in the near future. In particular, this cannot be expected where severely resource-constrained devices, e.g., IoT nodes, are involved. Hence, the need for appropriate testing concepts and tools apparently still persists.

Motivated by this, we present our work on designing and implementing a novel test methodology that aims to help developers and users to identify issues in black-box TLS implementations. Our work has been inspired by *Frankencerts* [22], an approach applying fuzzing techniques and differential testing to the X.509 certificate validation routines that the authentication in TLS is based on. In contrast to *Frankencerts*, however, we put our focus on the implementation of the TLS handshake protocol itself.

Briefly explained, we use semi-randomly generated TLS protocol messages to stimulate multiple TLS implementations with equivalent input and use discrepancies in their responses to detect implementation bugs. Our approach does neither require code instrumentation nor execution monitoring, a property particularly interesting for developers of embedded platforms. At the heart of our approach is a novel fuzzing algorithm that generates TLS protocol messages which are highly effective in triggering discrepant protocol behavior among different TLS implementations.

Clearly, differential testing itself is not a new concept [23]. However, to the best of our knowledge, we are the first to apply fuzzing-based differential testing to the TLS handshake protocol itself. In previous work, differential testing has only been applied to the validation of certificates [22], [24], [25], [26].

We see several use cases for our approach:

- *Debugging and nonconformance detection*: Comparing a set of different TLS implementations among each other or comparing an implementation under test

against a reference implementation is a way to detect implementation bugs. Moreover may it expose behavior that does not conform to the specification.

- *Regression testing*: Two versions of the same protocol implementation might be contrasted in order to gain confidence that, while changes and improvements have been implemented, no unintended change of behavior has been introduced.
- *Fingerprinting*: Comparing an unknown remote protocol implementation to a set of known reference implementations might allow to infer the origin of the unknown protocol implementation.
- *Identifying imprecise standardisation*: When comparing a set of protocol implementations, disagreement among the implementations may serve as a pointer to an imprecise protocol specification.

This paper presents the following contributions:

- *Generic Message Trees*: We design and implement the concept of *Generic Message Trees* (GMTs), a versatile and dynamic data structure for efficiently operating on highly-structured protocol messages.
- *Fuzzing-based TLS message generation*: Based on the GMT concept, we present a randomized algorithm for generating highly diverse and *mostly-valid* TLS handshake messages. Mostly-valid means that messages tend to obey all except very few syntactic and semantic rules.
- *Differential testing of ClientHello parsing*: We use our message generation algorithm to differentially test five popular TLS server implementations: *OpenSSL*, *BoringSSL*, *WolfSSL*, *mbedtls*, and *MatrixSSL*. In that course, we show that our algorithm is more effective in provoking discrepant responses than American Fuzzy Lop [27], *TLS-Attacker* [28], and *NEZHA* are. We study some of the bugs we exposed in TLS implementations using our test approach.
- *Software framework*: We make our software available for other researchers and developers [29].

Currently, our approach is limited to challenging the parsing routines for ClientHello messages embedded in TLS server implementations. Note that during this initial phase of the TLS handshake cryptography does not pose a challenge. However, an important goal of our ongoing work is to overcome these limitations and allow to test the whole handshake process, including the client perspective.

Our paper is organized as follows: Section 2 gives an overview of the TLS protocol, while Section 3 outlines related work. The concept of GMTs is presented in Section 4. Section 5 explains our test methodology, while its evaluation is provided in Section 6. A study of some TLS implementation bugs that we found is given in Section 7. Finally, Section 8 sketches future research directions and Section 9 summarizes the paper.

2 THE TLS PROTOCOL FAMILY

The TLS protocol provides a flexible framework for cryptographically securing communication from end to end over networks assumed to be under an active attacker's control. Since its outset, TLS evolved from version 1.0 [30] to its currently latest version 1.2 [1]. At present, version 1.3 of TLS is under preparation [31].

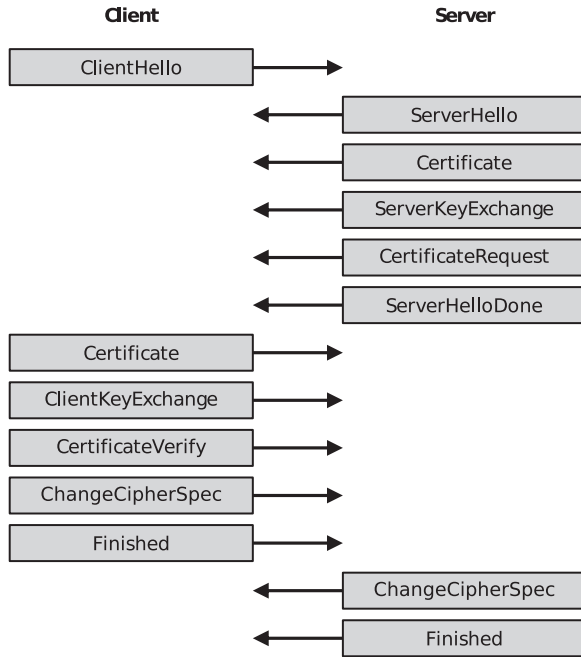


Fig. 1. Illustration of the sequence of TLS protocol messages typically exchanged between a TLS client and a TLS server during a handshake with mutual authentication. Depending on the authentication and key agreement scheme, some message may be optional or have a context-dependent interpretation. Note that with the upcoming TLS 1.3 the handshake is going to change significantly [31].

TLS is composed of five sub-protocols. The *Record Protocol* fragments data from higher TLS layers and handles data encryption and integrity protection. Its protocol data unit is called a *TLS record*. Stacked on top are the *Handshake*, *Change Cipher Specification*, *Alert*, and *Application Data* protocols. The first two are responsible for negotiating connection parameters and cryptographic keys. The Alert Protocol delivers error messages and the Application Data Protocol is used to securely transfer application data.

A TLS connection is established via a handshake where client and server exchange multiple messages (see Fig. 1). Each handshake message comprises a complex structure of binary-encoded integer, enumeration, and opaque fields as well as nested sub-structures. A client initiates a TLS handshake by sending a *ClientHello* message with a set of supported protocol options. If possible, the server responds with a *ServerHello* message and conveys its respective choice of options. Otherwise, the server typically responds with an Alert messages.

Authentication of the server (and optionally the client) as well as the establishment of a shared cryptographic secret is achieved by exchanging *Certificate* and *KeyExchange* messages. Finally, *Finished* messages complete the handshake and confirm its authenticity in retrospect by means of a signature over a transcript of the full handshake. Several extension points in the messages allow to retrofit new protocol features.

3 RELATED WORK

Various generic black-box approaches use a model of the system under test—either learned from dynamic interactions or provided as additional input—to find vulnerabilities in

network protocol implementations [32], [33], [34], [35], [36]. However, the applicability of such approaches tends to scale poorly with the complexity of the protocol under consideration; a fact that seems prohibitive for TLS.

De Ruiter and Poll use state machine learning to infer high-level states and transitions in black-box TLS implementation [37]. Based on visualizations of the learned state machines bugs are identified via manual investigation. Similarly, Beurdouche et al. systematically test TLS implementations for state machine bugs by checking whether an invalid protocol flow is accepted [38]. Invalid protocol flows are generated by skipping or repeating messages, or by hopping between message traces from different protocol modes. However, both approaches treat protocol messages as atomic units and do not challenge corresponding parsing routines.

TLSFUZZER [39] is a TLS test suite allowing to generate offensive TLS protocol flows with invalid messages. The expected implementation behaviour has to be specified in the test definitions explicitly; automated detection of incorrect implementation behavior does not seem to be supported.

Somorovsky presented a framework called *TLS-Attacker* [28], [40]. Besides testing for the susceptibility to known cryptographic attacks, it allows evaluating TLS server behavior by using a fuzzing approach based on dynamic modifications of message fields. *TLS-Attacker* detects implementation issues using instrumentation-based runtime monitoring of the target or by using a TLS context analyzer that investigates the correctness of TLS protocol flows.

Following the notion of differential testing [23], *SFADiff* is a generic approach based on *Symbolic Finite Automata* learning to systematically derive differences between a set of programs [41]. *SFADiff* has been evaluated successfully in three different settings, though none of them in the context of TLS. However, automata learning becomes prohibitively complex for TLS unless protocol interactions use a very limited alphabet.

Frankencerts [22] and *Mucerts* [24] apply differential testing to the certificate validation logic in TLS implementations. Where the latter disagree on the validity of one and the same randomly mutated X.509 certificate, a manual investigation of the root cause has to follow. However, neither approach addresses the TLS handshake itself.

Sivakorn et al. presented *HVLearn*, a black-box approach for testing hostname verification as part of TLS certificate validation [26]. From each implementation *HVLearn* infers a model that describes the set of all hostnames that match a given certificate template. Bugs in a model are detected by finding discrepancies with models of other implementations or by checking against regular-expression-based rules derived from the specification.

Petsios et al. presented *NEZHA*, an approach for domain-independent differential testing [25]. *NEZHA* uses the evolutionary fuzzing engine *libFuzzer* [42] guided by various metrics that specifically target differential testing. Applied to the certificate validation in TLS implementations, *NEZHA* finds significantly more discrepancies than *Frankencerts* and *Mucerts*.

We observe that in the context of TLS, differential testing has only been applied to the certificate validation (or parts

```

16 03 01 01 2C 01 00 01 28 03 03 40 C7 0E 24 30 01 B9 6D 8C 63 68 77 38 69 64 32 D3 E6 F9 49 10 7A AB AD 84 50 CD FF D6 A2 66 E4 00 00 92 C0 30 ...

{0:TLSRecord}          TLSRecord
|--[0:type]            RecordType          | 16                | handshake (22)
|--[1:version]         ProtocolVersion     | 03 01            | TLSv1 (769)
|--[2:length]          Integer             | 01 2C            | 300
--[3:msg]              HandshakeMessage
  |--[0:type]          HandshakeType       | 01                | client_hello(1)
  |--[1:length]        Integer             | 00 01 28         | 296
  --[2:msg]            ClientHello
    |--[0:version]     ProtocolVersion     | 03 03            | TLSv1_2 (771)
    |--[1:random]      OpaqueBlob          | 40 C7 0E 24 30 01 B9 6D
    |                  | 8C 63 68 77 38 69 64 32
    |                  | D3 E6 F9 49 10 7A AB AD
    |                  | 84 50 CD FF D6 A2 66 E4
    |--[2:session_id_length] Integer         | 00                | 0
    |--[3:session_id]   OpaqueBlob          |
    |--[4:cipher_suites] ClientHello_cipher_suites
    |   |--[0:N]         Integer             | 00 92            | 146
    |   |--[1:V]         DynamicVector      |
    |   |--[0:CipherSuite] CipherSuite       | C0 30            | TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (49200)
    ...
  
```

Fig. 2. Illustration of the raw data representation (octet string in hexadecimal notation above the horizontal line) as well as the GMT representation (tree structure below the horizontal line) of the first part of a TLS *ClientHello* message (adopted from [43]). GMT nodes are printed in the same order as they are traversed for serialization. The four columns of the GMT’s visualization reflect (from left to right) the tree structure, the nodes’ types, the leaf nodes’ raw data representations, and a human-readable representation (where applicable), respectively.

thereof). Moreover do all previous approaches targeting TLS seem to involve developing dedicated source code that can handle the large variety of TLS message types. Not only does it significantly add to the complexity and the susceptibility to bugs of the test tools themselves. It also makes the tools’ maintenance laborious and binds the test quality more than necessary to the developer’s thoroughness. In the following Section 4, we present an approach that allows a reduction of protocol-specific implementation efforts.

4 GENERIC MESSAGE TREES

Our test approach is based on interacting with TLS implementations via an exchange of TLS protocol messages. That is, at the very least we need to generate suitable and sufficiently diverse TLS messages for stimulation. The large diversity and complexity of TLS protocol messages poses a considerable challenge here. Moreover do we need to generate *invalid* messages just as we need to generate *valid* ones. These prerequisite can often only be met by dedicated and specifically implemented message handling routines.

For this purpose, we propose the concept of *Generic Message Trees*. GMTs constitute an integral part of our methodology. In fact, it is the GMT concept that allows our methodology to be aware of the TLS protocol message syntax while being protocol-agnostic to the greatest possible extent.

4.1 The GMT Concept

The concept of GMTs is closely related to the concept of parse trees, but adds means e.g., for *in-place* message generation and manipulation. A GMT is an ordered rooted tree with typed nodes, which each represent a particular message component. A leaf node represents an atomic message field (e.g., an integer field) that allows a direct translation from and to its raw (*on-the-wire*) data representation. An internal node represents a composite message component whose content is recursively given by its child nodes.

Throughout this paper, we use the following notation. Let \mathcal{T} denote the set of all finite GMTs (including the empty GMT \emptyset with zero nodes), and, given some GMT $t \in \mathcal{T}$, let

\mathcal{V}_t denote the set of all nodes of t . Each node $v \in \mathcal{V}_t$ defines a subtree $\hat{v} \in \mathcal{T}$ that consists of node v and all its descendants. Furthermore, let \mathcal{S} be the set of all finite octet (byte) strings. The length (number of octets) of an octet string $s \in \mathcal{S}$ is referred to as $|s|$.

As an example, Fig. 2 shows an excerpt of the GMT representation of a TLS *ClientHello* message.

4.2 Dissection and Serialization

Dissection denotes the process of translating the raw data representation $s \in \mathcal{S}$ of some message into its GMT representation $t \in \mathcal{T}$. Dissection can be expressed as a function $\mathbf{D} : \mathcal{S} \rightarrow \mathcal{T}$. The reverse operation is called *serialization* and is achieved by traversing the GMT depth-first and concatenating the raw data representation of each leaf node $v \in \mathcal{V}_t$. Serialization can be expressed as a function $\mathbf{S} : \mathcal{T} \rightarrow \mathcal{S}$.

Observe that by their definition, GMTs are an inherently generic concept. Most format-specific aspects are essentially abstracted by the dissection function \mathbf{D} . Therefore, \mathbf{D} is highly dependent on the format definition of the message to be dissected. This is in contrast to the serialization function \mathbf{S} , which is fully generic.

4.3 Deriving Dissection Functions

Obviously, the benefits of GMTs were small if there was no efficient and automated way to obtain implementations of \mathbf{D} . Our approach to achieve this is based on the *TLS Presentation Language* (TPL). TPL is used more or less consistently throughout the TLS specification for the purpose of defining the content and encoding of TLS protocol messages [1]. TPL comprises basic message fields (i.e., integer and enumeration fields) as well as lists (also referred to as *vectors*) of fields, constructed (composite) types, and variants (i.e., dynamic choices within constructed types). Message definitions in TPL use a syntax quite close to that of the “C” programming language.

In the light of the aforementioned, TPL presents itself as an interesting candidate for automatically deriving implementations of \mathbf{D} . However, as TPL suffers from a somewhat casual and informal definition, for our purposes we use

eTPL, an enhanced version of TPL [43]. *eTPL* features an improved expressiveness and is suitable for an automated generation of parsers, i.e., implementations of \mathbf{D} , and other format-specific routines. As a consequence, the task of manually implementing TLS-specific details can be minimized. It essentially boils down to copying the TPL-based message definitions from the TLS specification and using *eTPL* constructs to complement the definition by those features of the message format that TPL fails to cover.

4.4 GMT Operators

Given the generic nature of GMTs we define a *GMT operator* O as an abstract manipulation agent acting on a GMT t . An operator O is invoked on a certain GMT node $v \in \mathcal{V}_t$, but may affect any node in the GMT. For example, the node an operator is invoked on may merely serve as a starting point for sophisticated operations walking the whole GMT.

Not every operator is applicable to every node in a meaningful way. Thus, each operator O comes with a *filter function* $F_O : v \mapsto f \in \{0, 1\}$, which returns 1 if operator O may be invoked on node v , and 0 otherwise.

Observe that certain operations on a GMT or its nodes change the GMT in such a way that redissecting it (i.e., dissecting the tree's serialization) yields a structurally different GMT, i.e., $\mathbf{D}(\mathbf{S}(t)) \neq t$. Such *redissection instability* of a GMT is primarily caused by a violation of dependencies between message fields (e.g., out-of-bounds lengths).¹

5 METHODOLOGY

As the integral component of our methodology for testing implementations of TLS—the handshake in particular—we design a fuzzing algorithm that is capable of generating highly diverse corpuses of test messages. Our algorithm is designed such that the generated messages are very effective in provoking response discrepancies among different TLS implementations. In the spirit of differential testing, we then use such discrepancies as means to identify bugs in the implementations under test.

Actually, fuzz testing typically involves monitoring the test target for evidence of faulty behavior; for example using code instrumentation or dynamic memory monitoring. However, such measures may not be possible, e.g., because no source code is available. Moreover may some bugs, e.g., semantic bugs [44], not become manifest in the form of program crashes or invalid memory accesses. Therefore, combining differential testing with fuzz allows to detect less obvious failure cases.

With our methodology we try to close the gap between model-based approaches [23], [32], [33], [34], [35], [36], [37]—for which TLS is prohibitively complex, unless protocol interactions are significantly simplified—and approaches that are based on explicit test oracles [28] or hand-crafted protocol interactions [39]. In order to support settings that do not allow code instrumentation or dynamic memory monitoring, we intentionally refrain from using techniques in the design of our methodology that would break its black-box feature.

1. Redissection instability turns out to be a property worth considering when tracing the code path an independent parser is following while processing a GMT's serialization.

In the following, we use the term *implementation peer group*, or simply *peer group*, to refer to the set of protocol implementations that receive the same stimulation input and of which responses are compared among each other. *Test corpus* refers to a set of individual TLS messages which are used to stimulate implementations in the peer group.

5.1 Input Generation

Our fuzzing algorithm is based on the manipulation of template TLS messages. To this end, it makes use of the concept of GMTs and corresponding manipulation operators as introduced in the previous Section 4. In contrast to other black-box fuzzing approaches we are aware of, our approach allows to cover the space of mostly-valid TLS messages with minimal implementation effort, yet very high efficiency.

5.1.1 Manipulation Operators

We define the following set of *deterministic* operators. Where applicable, v' refers to node v after the operator application.

- *Voiding operator* O_{void} : Replace subtree \hat{v} by an empty leaf node. This operation effectively removes all contributions of \hat{v} to the GMT's serialization, but retains a leaf node as a place holder potentially important for subsequent operations.
- *Removing operator* O_{rem} : Fully remove subtree \hat{v} . The effect of this operator on the serialization of node v is similar to that of O_{void} .
- *Duplicating operator* O_{dupl} : Duplicate subtree \hat{v} and insert the copy as a sibling, i.e., as a child of v 's parent to the right of v .

Additionally, we define several fuzzing operators.

- *Truncating fuzz operator* $O_{\text{trunc}}^{\text{fuzz}}$: Remove and truncate the nodes of subtree \hat{v} such that its serialization after the operator's application is truncated to length n , i.e., $|\mathbf{S}(\hat{v}')| = n$. The length n after truncation is chosen uniformly at random between zero and $|\mathbf{S}(\hat{v})| - 1$.
- *Integer fuzz operator* $O_{\text{int}}^{\text{fuzz}}$: Set the integer i represented by leaf node v to a new value i' . In *full range mode*, i' is chosen uniformly at random between zero and $2^w - 1$, where w is the integer field's bit width. In *proximity mode*, i' is chosen uniformly at random between zero and $2 \cdot i + 1$. Proximity mode helps keeping integers close to typical values. For each application of $O_{\text{int}}^{\text{fuzz}}$, full range mode or proximity mode are chosen at random with equal probabilities. This operator is only applicable to leaf nodes representing integers.
- *Content fuzz operator* $O_{\text{cont}}^{\text{fuzz}}$: Randomize the raw data representation of leaf node v . The length is either left unchanged (*update-only mode*) or chosen uniformly at random between zero and $2|\mathbf{S}(\hat{v})| + 1$ (*resize-and-update mode*). For each application of $O_{\text{cont}}^{\text{fuzz}}$, the choice between update-only mode and resize-and-update mode is made at random with equal probabilities. For $|\mathbf{S}(\hat{v})| = 0$ resize-and-update mode is chosen with a probability of one. This operator is only applicable to leaf nodes.
- *Appending fuzz operator* $O_{\text{app}}^{\text{fuzz}}$: Insert a new leaf node with randomized content as a child of node v . The raw data length is chosen uniformly at random

```

1: procedure REPAIR( $v$ , randomize)
2:   while  $v \neq \perp$  do           ▷ Repeat until done with root
3:     if not randomize or RND({true, false}) then
4:       REPAIRLOCAL( $v$ )
5:     end if
6:      $v \leftarrow$  PARENTOF( $v$ )   ▷ One step towards root
7:   end while
8: end procedure

```

Fig. 3. The repair algorithm used for restoring consistency between nodes within a GMT. Starting from node v , the algorithm moves towards the GMT’s root. In deterministic mode ($\text{randomize} = \text{false}$), for each node on the path towards the root local inconsistency among each node’s direct children are resolved using a node-specific function REPAIRLOCAL. In randomized mode ($\text{randomize} = \text{true}$), nodes are skipped randomly with a probability of one half (RND function call). RND(X) returns an element $x \in X$ uniformly drawn at random from the set X . PARENTOF(v) returns the parent node of v or \perp if v is the root node.

between one and four. This operator is only applicable to internal nodes.

- *Synthesizing fuzz operator* $O_{\text{syn}}^{\text{fuzz}}$: Replace the subtree \hat{v} by a semi-randomly synthesized subtree that obeys the message syntax inherent in the type of node v . Despite synthesizing *syntactically* correct message, $O_{\text{syn}}^{\text{fuzz}}$ is not aware of the corresponding *semantics*.

Note that, conceptually, $O_{\text{syn}}^{\text{fuzz}}$ is the most powerful operator. Based on the dissection functions derived from eTPL definitions and implanted in GMT nodes, it is capable of synthesizing message components that are not present in the template message. In this way, any TLS extension may be synthesized in a syntactically correct way.

5.1.2 Resolving Inconsistencies (Repairing)

The application of certain manipulation operators is likely to destroy the internal consistency of a GMT. For instance, when a variable-length message component is truncated, the corresponding length field should be updated.

Obviously, there is a trade-off between intentionally generating inconsistent messages (trying to challenge an implementation’s input validation routines) and avoiding message inconsistency (trying to prevent an early rejection of invalid messages).

Therefore, we define a repair algorithm resolving inconsistency among the nodes of a GMT on a semi-random basis (see Fig. 3). The algorithm proceeds by starting at some node v and traversing the GMT towards its root. For each node on this path, local inconsistencies among the node’s direct children are resolved using a node-specific repair routine.² If run in randomized mode, nodes are skipped randomly with a probability of $p = 1/2$, giving rise to incidental consistency violations within the GMT.

5.1.3 Input Generation Algorithm

The core of our algorithm for TLS message generation is presented in Fig. 4. It starts from the GMT representation of a template message and repeatedly applies a

2. These repair routines are automatically derived from the eTPL-based message definitions just as the dissection functions are.

randomly chosen manipulation operator to a suitable, randomly selected GMT node. After each operator application (excluding $O_{\text{int}}^{\text{fuzz}}$), the repair algorithm is invoked on the node to which the previous operator has been applied. The number of operator applications is implicitly chosen at random; after each iteration the algorithm stops with a probability of $p = 1/2$. The serialization of the final GMT is the message to be used for stimulation.³

In order to avoid the generation of bit-identical duplicates, we keep a list of the hashes of previously generated messages. Having said that, this list is the only state our generation algorithm holds.

Unfortunately, unless recording internet traffic at large scale, obtaining a large corpus of TLS template messages is a nontrivial problem. As we show in Section 6, our algorithm turns out to be effective even if only a single template messages is provided.

5.2 Response Analysis

In the following, we describe the differential analysis of responses received from the implementations in the peer group upon stimulation.

5.2.1 Response Reduction

As a consequence of testing TLS protocol implementations, the responses we have to deal with are complex TLS protocol messages that lack straightforward comparability. For differential testing, it is therefore crucial to define a reasonable metric that can assess whether responses are discrepant or in agreement—potentially despite differences in their raw data representations. We abstract the concrete choice of this metric by using a *reduction function* R that maps a TLS implementation’s response, represented as a GMT $t \in \mathcal{T}$, to a *reduced response set* \mathcal{R} , i.e., $R : \mathcal{T} \rightarrow \mathcal{R}$. Responses $t_A, t_B \in \mathcal{T}$ from two implementations A and B are considered in agreement under reduction function R if $R(t_A) = R(t_B)$ and discrepant otherwise.⁴

In the following, let \mathcal{P} denote the implementation peer group and let \mathcal{X} refer to a test corpus, i.e., a set of stimuli $x \in \mathcal{X}$. Furthermore, let $t_{i,x}$ denote the GMT representation of the response obtained from implementation $i \in \mathcal{P}$ when stimulated with stimulus $x \in \mathcal{X}$. We define the *response signature* as the tuple of implementation responses reduced under some R as

$$T_R(\mathcal{P}, x) = \langle R(t_{1,x}), \dots, R(t_{|\mathcal{P}|,x}) \rangle. \quad (1)$$

Clearly, we are most interested in cases where a *single* stimulus induces *discrepant* responses in the peer group. As an indicative measure, we define the number of *unique responses* obtained for a single stimulus as

3. Note that most protocol-specific dependencies of our algorithm are encapsulated in the corresponding GMT dissection and repair routines.

4. It is clear that the choice of a reduction function R affects both the sensitivity to real implementation issues as well as the susceptibility to innocuous discrepancies. Optimizing the choice of R is out of the scope of this paper though.

```

1: function FUZZ( $v_0$ )                                     ▷ Fuzz  $v_0$ , the subtree whose root is  $v_0$ 
2:   repeat
3:      $o \leftarrow \text{RND}(\{\mathcal{O}_{\text{void}}, \mathcal{O}_{\text{rem}}, \mathcal{O}_{\text{dupl}}, \mathcal{O}_{\text{trunc}}^{\text{fuzz}}, \mathcal{O}_{\text{int}}^{\text{fuzz}}, \mathcal{O}_{\text{cont}}^{\text{fuzz}}, \mathcal{O}_{\text{app}}^{\text{fuzz}}, \mathcal{O}_{\text{syn}}^{\text{fuzz}}\})$    ▷ Select an operator at random
4:      $v \leftarrow \text{RND}(\{v \in \mathcal{V}_{v_0} | F_o(v) = 1\})$        ▷ Select a suitable node in  $v_0$  at random
5:     APPLYOPERATOR( $o, v$ )                               ▷ apply operator  $o$  to node  $v$ 
6:     if  $o \neq \mathcal{O}_{\text{int}}^{\text{fuzz}}$  then                         ▷ Unless an integer field has been fuzzed ...
7:       randomize  $\leftarrow \text{RND}(\{\text{true}, \text{false}\})$ 
8:       REPAIR( $v, \text{randomize}$ )                            ▷ ... repair GMT (e.g. update length fields)
9:     end if
10:    if  $o \in \{\mathcal{O}_{\text{dupl}}, \mathcal{O}_{\text{syn}}^{\text{fuzz}}\}$  and  $\text{RND}(\{\text{true}, \text{false}\})$  then   ▷ Duplicated or synthesised subtrees ...
11:      FUZZ( $v$ )                                          ▷ ... may be fuzzed recursively
12:    end if
13:  until  $\text{RND}(\{\text{true}, \text{false}\})$                  ▷ The number of operator applications becomes approx. exponentially distributed
14:  return  $v_0$                                        ▷ Return the manipulated (sub)tree
15: end function

```

Fig. 4. The fuzzing algorithm used to semi-randomly generate mostly-valid TLS messages. The algorithm starts from the GMT representation v_0 of a template message and repeatedly applies a randomly selected manipulation operator to a randomly selected node. In-between operator applications message inconsistencies are resolved on a semi-random basis using the repair algorithm shown in Fig. 3. $\text{RND}(X)$ returns an element $x \in X$ uniformly drawn at random from the set X .

$$N_R(\mathcal{P}, x) = |\{R(t_{i,x}) | i \in \mathcal{P}\}|. \quad (2)$$

Obviously, $1 \leq N_R(\mathcal{P}, x) \leq |\mathcal{P}|$ with $N_R(\mathcal{P}, x) = 1$ if⁵ and only if all implementations in \mathcal{P} are in agreement with respect to R .

5.2.2 Identification of Root Causes

Once a response discrepancy has been detected, determining its root cause is a nontrivial problem. For the time being this remains as a manual task for the human tester. We sketch some ideas on how to address this challenge in Section 8 about future work.

Note that behavioral discrepancies may not only be due to implementation bugs. As alternative causes dissimilar features or configurations of the implementations as well as different interpretations of the specification need to be considered before attributing discrepant implementation behavior to a bug.

5.3 Capabilities and Limitations

In principle, our approach allows detecting any implementation issue that gets provoked by the stimulation input and then become manifest in an implementation's response behavior. That is, it does not require an implementation to crash or to perform an invalid memory access. Assuming the reduction function does not "map away" these deviations, an observable deviation of an implementation's output from those of the other implementations is sufficient. It is this fact that yields sensitivity to semantic bugs which might otherwise elude detection.

On the other hand, differential testing is generally blind with respect to defects that are present in all implementations in the peer group or that result in identical or indistinguishable (mis)behavior. Furthermore, silent issues that never affect the implementation's response behavior (e.g., memory leaks)

5. We deliberately ignore the somewhat degenerate case of an empty peer group here.

stay unnoticed, unless the stimulation provoking them lets other implementations behave observably different.

Currently, interactions with the TLS implementations under test are given by distinct and reiterate stimulus-response pairs. Our test system acts as a TLS client by repeatedly sending semi-randomly generated ClientHello messages over fresh transport connections (stimulus) and observing the TLS servers' responses. While this constitutes an obvious restriction (it only tests the ClientHello processing routines within TLS servers), we see our approach as presented herein only as the first step towards *fully interactive* differential testing of black-box TLS protocol implementations, covering both servers and clients as well as the full TLS handshake process. Dealing with the stateful nature of TLS and its use of cryptography constitutes the major challenge in that direction (see also Section 8 on prospects and future work).

6 EVALUATION

In this and the following Section 7 we report on our results obtained from applying our test approach to *OpenSSL 1.0.2h*, *BoringSSL*,⁶ *MatrixSSL 3.8.4*, *mbedtls 2.2.0*, and *WolfSSL 3.9.8*. That is, our peer group \mathcal{P} is given by these five implementations. The list includes implementations for both PC as well as embedded platforms. We refrain from using more than five implementations so as to keep the complexity of investigating response discrepancies at a manageable level.⁷

In order to evaluate the efficiency and effectiveness of our generation algorithm, we compare our approach to other methods, i.e., *American Fuzzy Lop* (AFL) [27], *TLS-Attacker* [28], and *NEZHA* [25].

6. The version we use corresponds to commit 78684e5b222645828ca302e56b40b9daf2b2d27 on branch 2883 of the official *BoringSSL* Git repository [45].

7. The complexity of telling apart interesting from benign response discrepancies suffers from scaling unfavorably with the size of the peer group \mathcal{P} . Addressing this scalability issue in the context of TLS is going to be the subject of future work.

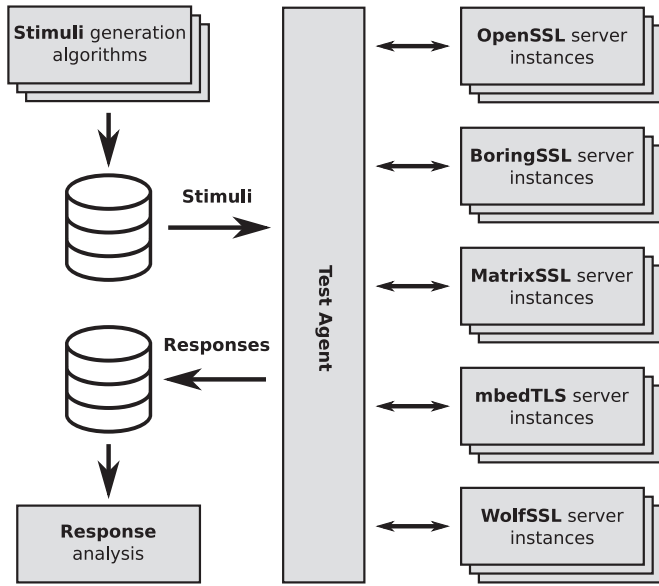


Fig. 5. Illustration of the experimental setup that we use to compare our test approach against AFL, *TLS-Attacker*, and *NEZHA*. Stimuli from different pre-generated test corpuses are used to stimulate five different TLS server implementations via their standard network interfaces. For each TLS implementation we launch 20 independent instances in order to speed up the test process through parallelization. Implementation responses are recorded for further analyses, but do not feed back to input stimulation. All components run on a single local machine.

6.1 Experimental Setup

Our experimental setup is sketched in Fig. 5. Using pre-generated test corpuses, we stimulate each TLS server implementation in our peer group and record corresponding responses for further analyses (see Section 6.1.2). Implementation behavior does not feed back to input stimulation. Additionally, we determine the code coverage in *MatrixSSL* 3.8.4 using *gcov* version 4.8.2 for each test corpus as a whole.

All components of our setup run on a single local machine (standard office PC, *Linux*-3.11-2 64 bit, Intel Core i5-4590, 8 GB RAM). TLS server implementations are compiled with *gcc* version 4.8.2 and use a similar compile and runtime configuration. The runtime configuration includes one and the same X.509 RSA server certificate and TLS 1.2 as the highest supported protocol version.

Communication with the TLS server implementations proceeds via the loopback network interface. To speed up the process, we parallelize test interactions by launching 20 independent instances of each server implementation in the peer group. In this way, our setup manages to handle on average approx. 55 stimuli (i.e., 55×5 server stimulations) per second using a per-instance response timeout of 100 ms.

6.1.1 Input Generation

For each generation method, i.e., our algorithm, *AFL*, *TLS-Attacker*, and *NEZHA* we build 100 independent test corpuses with 100'000 stimuli each.

As a start we use our generation algorithm described in Section 5.1.3 to generate a corresponding set of test corpuses. As template message we use one single ClientHello message with a length of 305 octets generated by *OpenSSL*'s command-line client (version 1.0.2h). The template ClientHello message offers TLS 1.2 as highest supported protocol version and lists 72 supported cipher suites. Furthermore, it contains five

different extensions: *Supported point formats* (with three entries) and *Supported Elliptic Curves* (with 25 entries) [46], *Supported signature algorithms* (with 15 entries), *Session ticket* (empty) [47], and *Heartbeat* [48]. Given these settings, our algorithm generates approx. 500 TLS messages per second.

Additionally, we use *AFL* version 2.40b run against an appropriately instrumented instance of *MatrixSSL* 3.8.4 to generate a similar set of independent *AFL* test corpuses. *AFL* is provided with the same single ClientHello message that we also use to seed our generation algorithm. We let *AFL* skip deterministic generation steps as we expect those to introduce a bias in the test corpus.⁸

Using the *TLS-Attacker* framework [28] we build a set of *TLS-Attacker* test corpuses based on the ClientHello messages emitted by *TLS-Attacker*. We run a slightly modified⁹ version of *TLS-Attacker* against *OpenSSL* 1.0.2h. Note that *TLS-Attacker* generates a large number of TLS protocol flows where ClientHello messages differ only in the *ClientRandom* field. In order to allow for a fair comparison, we therefore drop those quasi-duplicates before building a test corpus.

Finally, we let *NEZHA* [25] run against the server implementations in our peer group to generate a set of *NEZHA* test corpuses. We let *NEZHA* be guided by its output δ -diversity metric using distinct return codes for a ServerHello response, each different type of Alert message, and no response at all within a timeout of 100 ms.¹⁰

Note that, in contrast to our approach, *AFL*, *TLS-Attacker*, and *NEZHA* use life interactions with one or more TLS implementations to guide the input generation process. Nevertheless does our evaluation in all cases use the setup shown in Fig. 5 to consistently record implementation responses based on previously generated test corpuses.

6.1.2 Evaluation Metrics

As our primary evaluation metric, we define the number of *unique response discrepancies* Δ_R as

$$\Delta_R(\mathcal{P}, \mathcal{X}) = |\{T_R(\mathcal{P}, x) | x \in \mathcal{X} : N_R(\mathcal{P}, x) > 1\}|, \quad (3)$$

with $T_R(\mathcal{P}, x)$ as defined in Eq. (1) and $N_R(\mathcal{P}, x)$ as defined in Eq. (2). $\Delta_R(\mathcal{P}, \mathcal{X})$ is a measure of the capability of test corpus \mathcal{X} to induce distinct discrepancies among the peer group \mathcal{P} 's implementations.¹¹

Actually, we are not only interested in the *total* number of unique response discrepancies induced by a test corpus \mathcal{X} as a whole. Rather, we are mostly interested in *cumulative* numbers obtained from gradually increasing the number of stimuli taken from the test corpus. Therefore, by \mathcal{X}_n we denote a *truncated* test corpus that only consist of those n stimuli $x \in \mathcal{X}$ that have been generated first. We then use $\Delta_R(\mathcal{P}, \mathcal{X}_n)$ as a function of n with $0 \leq n \leq |\mathcal{X}|$.

8. *AFL* would run much more than 100'000 deterministic generation steps.

9. We added a few lines in the framework's source code that allow recording the generated ClientHello messages on disk. Our version is based on commit fd74472c9950c4415a88934a3c21808ac3b078d3 from *TLS-Attacker*'s official GitHub repository [40].

10. This reduction scheme is equivalent to the reduction function R_2 defined in Eq. (5) of the following Section 6.1.2.

11. Note that if the condition $N_R(\mathcal{P}, x) > 1$ were dropped, the definition of Δ_R would be identical to the *output δ -diversity* defined by Petsios et al. [25].

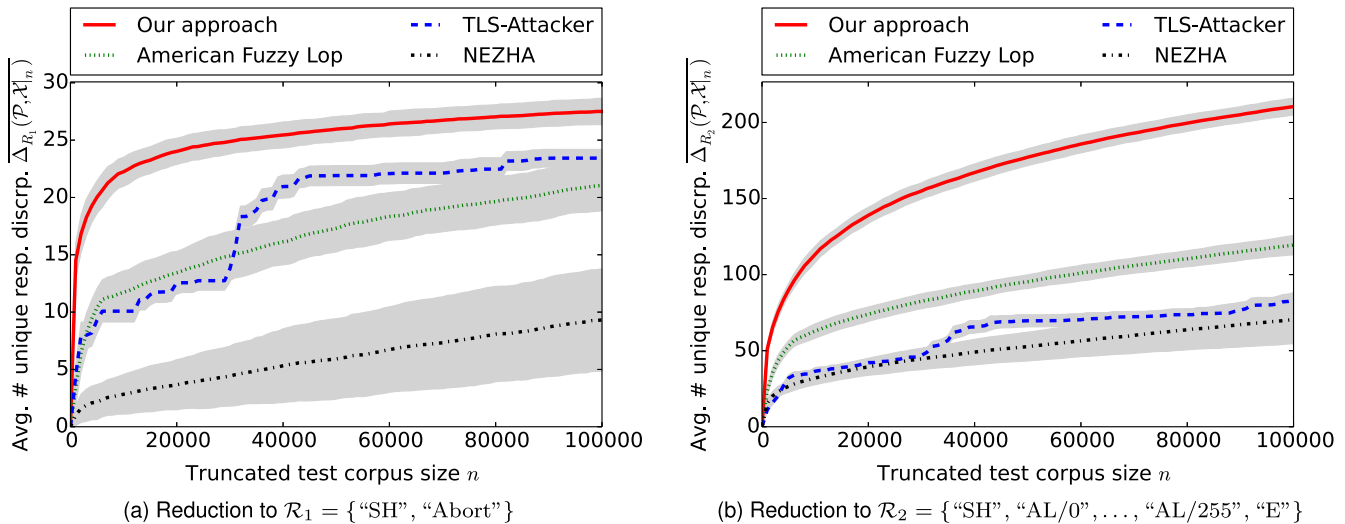


Fig. 6. The average number of unique response discrepancies $\overline{\Delta_R(\mathcal{P}, \mathcal{X}|_n)}$ for Δ_R as defined in Eq. (3), plotted as a function of the truncated test corpus size n . $\overline{\Delta_R}$ is plotted for test corpuses generated with our approach (red solid line), AFL (green dotted line), *TLS-Attacker* (blue dashed line), and *NEZHA* (black dash-dotted line). The average is taken for each generation method individually over the respective 100 test corpuses; the grey bands represent the corresponding standard deviation. The plot is given for the two reduction functions $R = R_1$ (left) and $R = R_2$ (right) defined in Eqs. (4) and (5), respectively.

In accordance with Section 5.2.1 we define two response reduction functions $R_{1,2} : \mathcal{T} \rightarrow \mathcal{R}_{1,2}$ that feature different output granularity. R_1 has a binary outcome with $\mathcal{R}_1 = \{\text{"SH"}, \text{"Abort"}\}$. It reflects whether or not a server responds with a ServerHello message within a timeout of 100 ms, i.e.,

$$R_1(t) := \begin{cases} \text{"SH"} & \text{if } t \equiv \text{ServerHello} \\ \text{"Abort"} & \text{otherwise.} \end{cases} \quad (4)$$

R_1 is particularly suitable to find those stimuli where all implementations except one refuse to continue the handshake. In such cases, the deviant implementation is most likely to miss an important input validation check.

As a more fine-granular option, R_2 additionally takes into account error information in case no ServerHello message is received within the 100 ms timeout. With

$$\mathcal{R}_2 = \{\text{"SH"}, \text{"AL/0"}, \dots, \text{"AL/255"}, \text{"E"}\},$$

we set

$$R_2(t) := \begin{cases} \text{"SH"} & \text{if } t \equiv \text{ServerHello} \\ \text{"AL/}x\text{"} & \text{if } t \equiv \text{Alert of type } x \\ \text{"E"} & \text{if } t = \emptyset \text{ (empty)}. \end{cases} \quad (5)$$

Note that $R_2(t) = \text{"E"}$ either if a server silently closes the connection on TCP level or if it does not reply at all within the 100 ms timeout.

As a second evaluation metric, we use the code (line) coverage determined in *MatrixSSL 3.8.4*. For each test corpus, the code coverage is determined over stimuli from the test corpus as a whole. Note that we restrict the coverage analysis to those modules in *MatrixSSL* that are responsible for parsing incoming TLS messages.¹²

12. Concretely, these are `sslDecode.c` (parsing TLS records), `hsDecode.c` (parsing handshake messages), and `extDecode.c` (parsing ClientHello and ServerHello extensions) from the *MatrixSSL* sources. The code considered amounts to 1,617 executable lines in total.

6.2 Results

In this section we present the result of our evaluation using the metrics presented in the previous Section 6.1.2.

6.2.1 Unique Response Discrepancies

Fig. 6 shows the average of $\Delta_R(\mathcal{P}, \mathcal{X}|_n)$ as defined in Eq. (3) and its standard deviation plotted individually for our approach, AFL, *TLS-Attacker*, and *NEZHA* as well as the two reduction functions $R = R_1$ and $R = R_2$ defined in Eqs. (4) and (5), respectively. For each generation method, the average is computed over the respective 100 test corpuses.

It turns out that our generation algorithm is more effective in provoking discrepant responses than AFL, *TLS-Attacker*, and *NEZHA* are. Considering test corpuses as a whole (untruncated), our approach on average generates 17 percent more (for $R = R_1$) and 76 percent more (for $R = R_2$) unique response discrepancies than the respective second most effective approach.

The advantage of our approach seems to be present at least for test corpuses not significantly larger than 100'000 stimuli. However, recall that our approach basically is stateless. Δ_R is therefore likely to run into saturation at some point. In contrast to that follow the other approaches either an evolutionary (AFL and *NEZHA*) or a partially deterministic strategy (*TLS-Attacker*), making the long-term behavior hard to predict. An illustrative example is the abrupt jump in Δ_R at $n \approx 30'000$ for *TLS-Attacker*.

Furthermore, it is a notable observation that in our study AFL clearly outperforms *NEZHA*. This is in contrast to what the authors of *NEZHA* found when applying *NEZHA* to the validation routines of X.509 certificates in TLS implementations [25]. We surmise that our observation is due to the fact that we use only one single TLS message to seed the generation process, whereas in the case of certificate validation AFL and *NEZHA* have been given 1,000 template certificates.

TABLE 1

Average Code (Line) Coverage Determined in *MatrixSSL 3.8.4*

Test approach	Average line coverage [%]
Our approach	26.6 ± 0.3
AFL	25.7 ± 0.4
NEZHA	24.0 ± 0.6
<i>TLS-Attacker</i>	21.9 ± 0.2

6.2.2 Code Coverage

Just as for Δ_{R_i} , the average code (line) coverage in *MatrixSSL 3.8.4* and its standard deviation is computed over the 100 test corpuses of each generation method individually.

The resulting numbers are given in Table 1. Reaching approx. 26.6 percent of the executable lines in the corresponding modules of *MatrixSSL 3.8.4*, test corpuses generated with our algorithm achieve a code coverage slightly higher than that achieved with any other evaluated approach (≤ 25.7 percent).

Recall that our evaluation is restricted to server stimulations via ClientHello messages. The fact that the achieved code coverage is considerably below 100 percent is due to most of the unreached lines in *MatrixSSL* being responsible for parsing incoming messages other than ClientHello messages.

7 A STUDY OF SOME IDENTIFIED BUGS

In the following, we briefly study some bugs in *MatrixSSL 3.8.4* and *WolfSSL 3.9.8* that we revealed with the help of our test approach. Among these bugs are two more or less independent ones in *MatrixSSL 3.8.4* that in combination we consider as a serious vulnerability.

In total, we identified 16 issues attributable to distinct root causes with varying impact on security or interoperability. We refrain from listing every single issue but discuss the essential aspects of our findings in Section 7.3. In all cases did we responsibly disclose our findings to the corresponding developers. In the meanwhile fixes for all serious and some minor bugs have been released.

For the rest of this section we drop the version numbers and simply use *MatrixSSL* and *WolfSSL* to refer to *MatrixSSL 3.8.4* and *WolfSSL 3.9.8*, respectively.

7.1 Mishandling of Length Fields

A particularly interesting—and apparently very frequent—class of issues in TLS implementations is related to an incorrect or inconsistent handling of the length fields of nested message components during parsing. As a result, the faulty parser may misinterpret the message content or may even be vulnerable to attacks.

Fig. 7 illustrates situations that can expose such parser misbehavior. In a well-formed message the length of a message component (the outer frame) is consistent with the lengths of its nested components (the inner frames). An *overflow* situation with *excess data* arises if the length of the outer frame exceeds the total length of its inner frames. In contrast to that does an *underflow* situation with *deficit data* arise if the total length of the inner frames exceeds the length of the outer frame. Explicitly designed to do so, our algorithm presented in Section 5.1 generates messages that

Outer frame

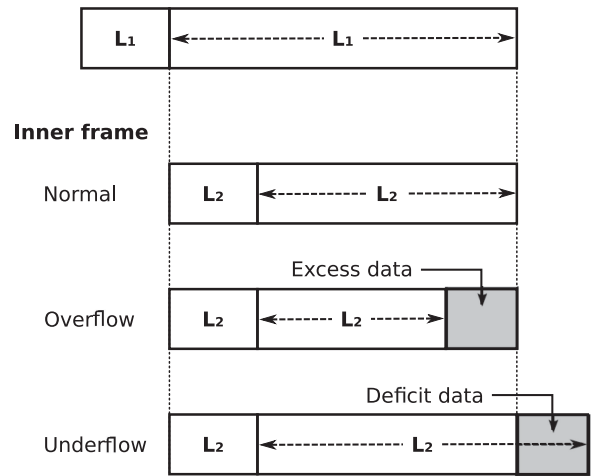


Fig. 7. Illustration of situations that may expose a message parser's mishandling of the length fields of nested message components (adopted from [43]). While normally redundant length fields are consistent, inconsistencies result in an *overflow* situation with real *excess data* (data provided by the outer frame of which the inner frame cannot make sense) or an *underflow* situation with *deficit data* (data the inner frame expects but the outer frame does not provide). In the absence of message fragmentation neither an overflow nor an underflow situation is expected or even acceptable.

randomly exhibit overflows and underflows situations in various message parts.

In the following sections, we briefly report on two corresponding bugs we found in *MatrixSSL* and *WolfSSL*.

7.1.1 *MatrixSSL: Unauthenticated ClientHello Extensions*

Because of its insufficient adherence to and inconsistent treatment of length fields when parsing ClientHello messages, *MatrixSSL* allows a *man-in-the-middle* (MITM) an unnoticed injection of ClientHello extensions into the TLS handshake between a *MatrixSSL* server and an arbitrary TLS client.

Concretely, on the one hand, *MatrixSSL* does not use the length specified in the header of an incoming handshake message to restrict the processing of a ClientHello message, but instead simply assumes that the end of the ClientHello message is aligned with the end of the record. On the other hand, *MatrixSSL* ignores the field specifying the length of the ClientHello extension list, assuming that the end of the extension list is aligned with the apparent end of the ClientHello message.¹³

Having said that, *MatrixSSL* does use the handshake header's length field to restrict what contributes to the handshake transcript. As illustrated in Fig. 8, this opens up an opportunity to a MITM to inject unauthenticated ClientHello extensions into any handshake with a *MatrixSSL* server by simply appending data to the TLS record containing the ClientHello message and only updating the record's length field.

Granting such power to a third party doubtlessly violates the assumptions that the security guarantees of TLS are based on. However, the severity of this vulnerability heavily

13. Note that because of the first issue, *MatrixSSL* effectively reads ClientHello extensions up to the end of the TLS record.

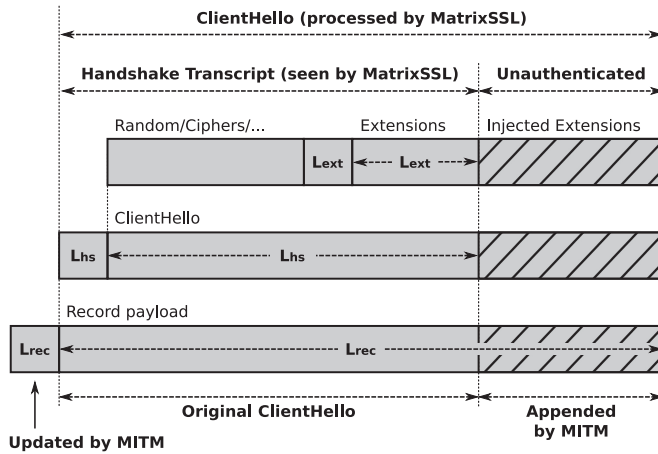


Fig. 8. Illustration of the vulnerability in *MatrixSSL* 3.8.4 that allows a man-in-the-middle (MITM) attacker to inject unauthenticated ClientHello extensions into a handshake between a *MatrixSSL* server and an arbitrary TLS client without either side noticing. The attacker simply appends data to the TLS record and only updates the length in the record's header. The vulnerability is a result of an incorrect/inconsistent treatment of length fields in the ClientHello processing routines of *MatrixSSL* 3.8.4. It allows to make the server process an extended ClientHello message while only the original ClientHello message contributes to the handshake transcript.

depends on the capabilities of the injected extensions. Considering the list of extensions supported by *MatrixSSL*, we currently do not see an obvious way to exploit this vulnerability. This assessment is *not* based on a thorough analysis, though.

7.1.2 WolfSSL: Erroneous Processing of Excess Data

Similarly to *MatrixSSL*, *WolfSSL* does not fully respect the length of a ClientHello message as specified in the handshake header. When parsing a ClientHello message, excess data within the ClientHello message (i.e., data following the ClientHello extension list) is treated as the beginning of a subsequent handshake message. Though this bug in *WolfSSL* does not seem to induce a security issue, it certainly is in conflict with the TLS specification.

Note the difference to the corresponding issue in *MatrixSSL*. While *WolfSSL* treats excess data as if it belongs to a *lower* level of nesting (the record layer), *MatrixSSL* treats excess data as if it belongs to a *higher* level of nesting (the extension list).

7.2 Faulty Parameter Negotiation

In the course of a TLS handshake, a client typically proposes a list of protocol parameter options to the server, from which the server then makes a choice of its preference. Lacking a mutually supported option, the server actively aborts the handshake by sending an appropriate alert message. This mechanism applies e.g., to cipher suites, compression methods, and other parameters. Note that, while still selected dynamically, the protocol version is negotiated in a different way.¹⁴

We give two examples of flawed negotiation implementations in the following.

14. Actually, with TLS 1.3 negotiation of the protocol version is going to be harmonized with the usual negotiation mechanism as described before [31].

7.2.1 MatrixSSL: Protocol Version Downgrade

Under certain circumstances, a *MatrixSSL* server does not select the highest TLS protocol version available between the client and the server. If the protocol version field in a ClientHello message (the field which informs the server about the highest version the client supports) is set to a version beyond TLS 1.2 (e.g., 0x0304 corresponding to TLS 1.3), the *MatrixSSL* server selects TLS 1.1, even though TLS 1.2 is supported by both client and server.

The security impact of this deviation from the specification currently seems negligible.¹⁵ However, it might become relevant when TLS 1.2 turns out to be insecure in a way that an attacker can affect the handshake between legitimate parties.

7.2.2 Nonobservance of Supported Compression Methods

The server implementation of *MatrixSSL*, *WolfSSL*, and *mbedtls* 2.2.0 do not consider the list of compression methods offered by the client in the ClientHello message when making the corresponding choice. Rather, these implementations select the *no compression* option even if not listed by the client.

There is no impact on security resulting from this (mis) behavior.

7.3 Discussion

As indicated before, the issues described in the previous sections are not the only we found.

For instance, there are several further cases where TLS implementations fail to systematically validate the length fields of nested message components. This class of implementation flaws therefore presents itself as a quite common one. To some extent we attribute this to the fact that only the draft version of the upcoming TLS 1.3 explicitly addresses these issues in a general statement.¹⁶ Previously released specifications are fairly faint in that respect as the general case is somewhat inauspiciously embedded in a very specific statement on ClientHello parsing.¹⁷

Presumably, further issues we found are due to the TLS specification not being very explicit in certain respects, too. Generally, it tends to be vague with regard to a receiver's liability to enforce rules it imposes on senders. This

15. TLS specification states that "If a TLS server receives a ClientHello containing a version number greater than the highest version supported by the server, it MUST reply according to the highest version supported by the server" [1].

16. The general statement is: "Peers which receive a message which cannot be parsed according to the syntax (e.g., have a length extending beyond the message boundary or contain an out-of-range length) MUST terminate the connection with a "decode_error" alert." [31].

17. Here, the TLS specification states: "A server MUST accept ClientHello messages both with and without the extensions field, and (as for all other messages) it MUST check that the amount of data in the message precisely matches one of these formats; if not, then it MUST send a fatal "decode_error" alert" [1]. Note that being very strict with regards to such validation checks is in a way in conflict with Postel's Law which demands: "Be conservative in what you do, be liberal in what you accept from others" [49]. However, in addition to the debatable applicability of Postel's Law to security protocols [50], [51], allowing superfluous excess data in protocol messages unnecessarily provides an attacker with extra means for affecting the handshake transcript and exploiting transcript collisions [28], [52].

effectively forces each individual implementor to balance the pursuits of robustness, security, and interoperability on his or her own.¹⁸ Moreover, in some cases the specification does not determine default parameters to be used when parameter negotiation as part of the protocol is incomplete. Fortunately, the current draft specification of the upcoming version 1.3 of TLS eliminates at least some of these deficiencies [31].

Note that, even though the immediate security impact of certain implementation misbehavior might be negligible, it can legitimately be considered undesirable from a meticulous point of view. In principle even subtle and seemingly benign misbehavior might eventually be turned into an attack-driving lever if used elaborately by a smart attacker [18], [53].

8 PROSPECTS AND FUTURE WORK

Our findings seem to indicate that differential testing applied to the TLS handshake protocol is a promising direction for improving the quality of TLS implementations. That said, several limitations of our current approach should and are going to be addressed in future work.

8.1 Fully Interactive Differential Testing

We consider the realization of fully interactive differential testing of the TLS handshake as the most interesting direction for future work. It would allow going beyond merely testing ClientHello processing in server implementations. However, the stateful nature of TLS and its use of cryptography present themselves as the major difficulties towards that. Nevertheless do we believe that harmonizing these conflicting aspects is possible.

The key idea is to introduce a TLS-specific and stateful *crypto filter* in the stimuli generation and response analysis loop. While the GMT representations of outgoing and incoming messages get passed through the crypto filter, its state (i.e., cryptographic algorithm selection, dynamically generated keys, etc.) is updated accordingly and cryptographic message components are inserted or removed, respectively. Thereby, the crypto filter essentially becomes the cryptographic TLS endpoint and keeps stateful aspects and cryptographic details away from the core algorithm. In order to allow for message manipulations before and after cryptographic computations, messages might be passed back and forth between the generation algorithm and the crypto filter. The efficiency of such a setting remains to be studied though.

With fully interactive testing comes a break-up of the current separation between stimuli generation and implementation stimulation. While removing the parallelizability feature of our current approach, it would allow for response-driven feedback to guide and dynamically optimize the generation algorithm (e.g., similar to the output δ -diversity guidance of *NEZHA*). Bringing in machine learning techniques at this point might be interesting as well.

18. As a potentially positive consequence, however, it enables implementation-specific optimizations where certain validation checks might be dropped for the benefit of reduced resource demands.

8.2 Further Research Directions

In addition to the aforementioned high-priority goal, there are further directions into which we plan to extend our work.

It is desirable to add support for further protocol versions, e.g., TLS 1.3, the upcoming successor of TLS 1.2 [31], and *Datagram TLS* (DTLS), the datagram version of TLS [54]. Moreover do we plan to add support for message fragmentation and coalescence at the TLS record layer.

Machine learning techniques might allow to simplify the task of manually correlating response discrepancies with root causes in the implementations. For instance, we plan to automatically derive specific characteristics in the stimulation messages that induce certain discrepancy patterns. In this way, the range of possible root causes might be narrowed down. Delta-debugging is another interesting approach in this direction [55].

Furthermore, as indicated previously, the long-term behaviour (i.e., for large test corpuses) of our approach should be studied and contrasted against that of others.

Finally, in the light of the high number of unique response discrepancies triggered by our stimulation messages, we expect our approach to be suitable for fingerprinting unknown TLS implementations [56]. We plan to study this potential in more detail.

9 CONCLUSION

In this paper, we presented our work on applying differential testing to black-box implementations of the TLS protocol. In contrast to previous work using differential testing in this context, we do not focus on the certificate validation logic of TLS implementations but directly address the TLS handshake protocol itself.

To this end, we presented a novel fuzzing strategy for efficiently generating *mostly-valid* TLS protocol messages. Our algorithm turns out to be highly effective: in our peer group with *OpenSSL*, *BoringSSL*, *MatrixSSL*, *mbedtls*, and *WolfSSL*, it induces more unique response discrepancies than *American Fuzzy Lop*, *TLS-Attacker*, and *NEZHA* do. Moreover does it achieve the highest code coverage in *MatrixSSL* among these alternative methods.

Compared to standard fuzzing, the differential test approach can achieve sensitivity to implementation bugs that do not express themselves in program crashes or invalid memory access. It allowed us to reveal a number of issues in popular TLS implementations. Among these is a serious vulnerability in *MatrixSSL 3.8.4* which a man-in-the-middle can use to inject unauthenticated ClientHello extensions into the TLS handshake between a *MatrixSSL* server and an arbitrary client. In all cases did we disclose our findings responsibly to the respective developers; fixes are available in the meanwhile.

While currently we use fuzz-generated ClientHello messages to stimulate TLS servers, we see our approach as presented herein only as the first step towards *fully interactive* differential testing of black-box TLS protocol implementations. In this context, dealing with the stateful nature of TLS and its use of cryptography is an obvious challenge. However, our ongoing work indicates that this problem is solvable.

Anyway, our findings let us think that applying differential testing to the TLS handshake is a promising direction for identifying bugs in TLS implementations. Our approach may serve as a valuable complement to existing test tools for TLS. In particular, it may allow developers of low-footprint, C-based TLS implementations to benefit from the promise of correctness related to verified implementations like *miTLS* [21] written in high-level languages.

Our software tools are available as open source projects at <https://github.com/hso-esk/tls-diff-testing>.

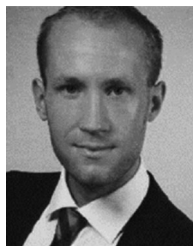
ACKNOWLEDGMENTS

We are very grateful for the valuable feedback we received from the anonymous reviewers of our manuscript as well as from our colleagues Manuel Schappacher, Artem Yushev, and Oliver Kehret. We furthermore would like to thank Prof. Dr. Dirk Timmermann from Rostock University for his scientific support of our work.

REFERENCES

- [1] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," RFC 5246, Aug. 2008, <https://www.rfc-editor.org/rfc/rfc5246.txt>
- [2] R. Oppliger, *SSL and TLS: Theory and Practice*. Norwood, MA, USA: Artech House, 2009.
- [3] L. C. Paulson, "Inductive analysis of the internet protocol TLS," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 3, pp. 332–351, Aug. 1999.
- [4] G. Díaz, F. Cuartero, V. Valero, and F. Pelayo, "Automatic verification of the TLS handshake protocol," in *Proc. ACM Symp. Appl. Comput.*, 2004, pp. 789–794.
- [5] K. Ogata and K. Futatsugi, "Equational approach to formal analysis of TLS," in *Proc. 25th IEEE Int. Conf. Distrib. Comput. Syst.*, Jun. 2005, pp. 795–804.
- [6] S. Gajek, M. Manulis, O. Pereira, A.-R. Sadeghi, and J. Schwenk, "Universally composable security analysis of TLS," in *Proc. 2nd Int. Conf. Provable Secur.*, Oct. 2008, pp. 313–327.
- [7] H. Krawczyk, K. G. Paterson, and H. Wee, "On the security of the TLS protocol: A systematic analysis," in *Proc. 33rd Annu. Cryptology Conf.*, Aug. 2013, pp. 429–448.
- [8] C. Meyer and J. Schwenk, "Lessons learned from previous SSL/TLS attacks - A brief chronology of attacks and weaknesses," Chair Network and Data Security, Ruhr University Bochum, Cryptology ePrint Archive, Report 2013/049, Jan. 2013, <https://eprint.iacr.org/eprint-bin/cite.pl?entry=2013/049>
- [9] Y. Sheffer, R. Holz, and P. Saint-Andre, "Summarizing known attacks on transport layer security (TLS) and datagram TLS (DTLS)," RFC 7457, Feb. 2015, <https://rfc-editor.org/rfc/rfc7457.txt>
- [10] Y. Sheffer, R. Holz, and P. Saint-Andre, "Recommendations for secure use of transport layer security (TLS) and datagram transport layer security (DTLS)," RFC 7525, May 2015, <https://rfc-editor.org/rfc/rfc7525.txt>
- [11] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov, "The most dangerous code in the world: Validating SSL certificates in non-browser software," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 38–49.
- [12] D. Kaloper-Mersinjak, H. Mehnert, A. Madhavapeddy, and P. Sewell, "Not-quite-so-broken TLS: Lessons in re-engineering a security protocol specification and implementation," in *Proc. 24th USENIX Secur. Symp.*, Aug. 2015, pp. 223–238.
- [13] "OpenSSL 'Heartbleed' vulnerability," *Common Vulnerabilities and Exposures*, 2013, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160>
- [14] "OpenSSL 'CCS injection' vulnerability," *CVE-2014-0224*, 2013, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0224>
- [15] OpenSSL: TLS/SSL and crypto library. (2017). [Online]. Available: www.openssl.org
- [16] "GnuTLS Certificate verification vulnerability," *CVE-2014-0092*, 2014, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0092>
- [17] GnuTLS: GNU Transport Layer Security Library. (2017). [Online]. Available: www.gnutls.org
- [18] S. Bratus, T. Darley, M. Locasto, M. L. Patterson, R. B. Shapiro, and A. Shubina, "Beyond planted bugs in 'Trusting Trust': The input-processing frontier," *IEEE Secur. Privacy*, vol. 12, no. 1, pp. 83–87, Jan./Feb. 2014.
- [19] D. A. Wheeler, "How to Prevent the next Heartbleed," Apr. 2014, <https://www.dwheeler.com/essays/heartbleed.html>
- [20] K. Bhargavan, C. Fournet, M. Kohlweiss, A. Pironti, and P. Y. Strub, "Implementing TLS with verified cryptographic security," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 445–459.
- [21] miTLS: A Verified Reference Implementation of TLS. (2017). [Online]. Available: mitls.org
- [22] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov, "Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 114–129.
- [23] W. M. McKeeman, "Differential testing for software," *Digit. Tech. J.*, vol. 10, no. 1, pp. 100–107, 1998.
- [24] Y. Chen and Z. Su, "Guided differential testing of certificate validation in SSL/TLS implementations," in *Proc. 10th Joint Meet. Found. Softw. Eng.*, Aug. 2015, pp. 793–804.
- [25] T. Petsios, A. Tang, S. J. Stolfo, A. D. Keromytis, and S. Jana, "NEZHA: Efficient domain-independent differential testing," in *Proc. 38th IEEE Symp. Secur. Privacy*, May 2017, pp. 615–632.
- [26] S. Sivakorn, G. Argyros, K. Pei, A. D. Keromytis, and S. Jana, "HVLearn: Automated black-box analysis of hostname verification in SSL/TLS implementations," *IEEE Symposium on Security and Privacy (SP)*, pp. 521–538, May 2017.
- [27] American fuzzy lop (AFL). (2017). [Online]. Available: <http://lcamtuf.coredump.cx/afl>
- [28] J. Somorovsky, "Systematic fuzzing and testing of TLS libraries," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1492–1504.
- [29] tls-diff-testing: Differential Handshake Fuzz-Testing of TLS Implementations. (2017). [Online]. Available: <https://github.com/hso-esk/tls-diff-testing>
- [30] T. Dierks and C. Allen, "The TLS protocol version 1.0," RFC 2246, Jan. 1999, <https://rfc-editor.org/rfc/rfc2246.txt>
- [31] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," draft-ietf-tls-tls13-20, Apr. 2017, Work in Progress, <https://tools.ietf.org/id/draft-ietf-tls-tls13-20.txt>
- [32] H. Gascon, C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck, "Pulsar: Stateful black-box fuzzing of proprietary network protocols," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2015, pp. 330–347.
- [33] G. Shu and D. Lee, "Testing security properties of protocol implementations - A machine learning based approach," in *Proc. 27th Int. Conf. Distrib. Comput. Syst.*, Jun. 2007, pp. 25–25.
- [34] G. Shu, Y. Hsu, and D. Lee, "Detecting communication protocol security flaws by formal fuzz testing and machine learning," in *Formal Techniques for Networked and Distributed Systems—FORTE 2008*, K. Suzuki, T. Higashino, K. Yasumoto, and K. El-Fakh, Eds. Berlin, Germany: Springer, 2008, pp. 299–304.
- [35] Y. Hsu, G. Shu, and D. Lee, "A model-based approach to security flaw detection of network protocol implementations," in *Proc. IEEE Int. Conf. Netw. Protocols*, Oct. 2008, pp. 114–123.
- [36] W. Tang, A.-F. Sui, and W. Schmid, "A model guided security vulnerability discovery approach for network protocol implementation," in *Proc. IEEE 13th Int. Conf. Commun. Technol.*, Sep. 2011, pp. 675–680.
- [37] J. de Ruiter and E. Poll, "Protocol state fuzzing of TLS implementations," in *Proc. 24th USENIX Secur. Symp.*, Aug. 2015, pp. 193–206.
- [38] B. Beurdouche, et al., "A messy state of the union: Taming the composite state machines of TLS," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 535–552.
- [39] H. Kario, "TLSPuzzer: TLS test suite and fuzzer," (2017). [Online]. Available: www.github.com/tomato42/tlspuzzer
- [40] TLS-Attacker: A Java-based Framework for Analyzing TLS Libraries. (2017). [Online]. Available: <https://github.com/RUB-NDS/TLS-Attacker>
- [41] G. Argyros, I. Stais, S. Jana, A. D. Keromytis, and A. Kiayias, "SFADiff: Automated evasion attacks and fingerprinting using black-box differential automata learning," in *Proc. 23rd ACM Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1690–1701.

- [42] libFuzzer: A library for coverage-guided fuzz testing. (2017). [Online]. Available: <http://lvm.org/docs/LibFuzzer.html>
- [43] A. Walz and A. Sikora, "eTPL: An enhanced version of the TLS presentation language suitable for automated parser generation," in *Proc. 9th IEEE Int. Conf. Intell. Data Acquisition Adv. Comput. Syst.: Technol. Appl.*, 2017, pp. 810–814.
- [44] L. Tan, C. Liu, Z. Li, X. Wang, Y. Zhou, and C. Zhai, "Bug characteristics in open source software," *Empirical Softw. Eng.*, vol. 19, no. 6, pp. 1665–1705, 2014.
- [45] BoringSSL: A fork of OpenSSL that is designed to meet Google's needs. (2017). [Online]. Available: boringssl.googlesource.com/boringssl
- [46] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller, "Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS)," RFC 4492, May 2006, <https://www.rfc-editor.org/rfc/rfc4492.txt>
- [47] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig, "Transport layer security (TLS) session resumption without server-side state," RFC 5077, Jan. 2008, <https://www.rfc-editor.org/rfc/rfc5077.txt>
- [48] R. Seggelmann, M. Tuexen, and M. Williams, "Transport layer security (TLS) and datagram transport layer security (DTLS) heartbeat extension," RFC 6520, Feb. 2012, <https://www.rfc-editor.org/rfc/rfc6520.txt>
- [49] J. Postel, "DoD standard transmission control protocol," RFC 761, Jan. 1980, <https://www.rfc-editor.org/rfc/rfc761.txt>
- [50] E. Allman, "The robustness principle reconsidered," *Commun. ACM*, vol. 54, no. 8, pp. 40–45, Aug. 2011.
- [51] M. Thomson, "The harmful consequences of Postel's maxim," *draft-thomson-postel-was-wrong-00*, Mar. 2015, Work in Progress, <https://tools.ietf.org/id/draft-thomson-postel-was-wrong-00.txt>
- [52] K. Bhargavan and G. Leurent, "Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Feb. 2016, <http://dx.doi.org/10.14722/ndss.2016.23418>
- [53] S. Bratus, M. E. Locasto, M. L. Patterson, L. Sassaman, and A. Shubina, (2011, Dec). Exploit programming: From buffer overflows to "Weird Machines" and theory of computation. *login.*, vol. 36, no. 6.
- [54] A. Freier, P. Karlton, and P. Kocher, "The secure sockets layer (SSL) protocol version 3.0," RFC 6101, Aug. 2011, <https://www.rfc-editor.org/rfc/rfc6101.txt>
- [55] A. Zeller and R. Hildebrandt, "Simplifying and isolating failure-inducing input," *IEEE Trans. Softw. Eng.*, vol. 28, no. 2, pp. 183–200, Feb. 2002.
- [56] C. Meyer, "20 years of SSL/TLS research: An analysis of the internet's security foundation," Ph.D. dissertation, Horst Görtz Inst. IT-Secur., Univ. Bochum, Bochum, Germany, Feb. 2014.



Andreas Walz received the diploma in physics from the University of Freiburg. Currently, he is working toward the PhD degree in the field of security in embedded systems with special interest in the TLS protocol family. From his studies and from working as a research assistant in experimental particle physics he has many years of experience in the development of embedded hardware systems as well as efficient software.



Axel Sikora received the diploma of electrical engineering and the diploma of business administration, both from Aachen Technical University, and the PhD degree in electrical engineering from the Fraunhofer Institute of Microelectronics Circuits and Systems, Duisburg, with a thesis on SOI-technologies. After positions in the telecommunications and semiconductor industry, he became a professor with the Baden-Wuerttemberg Cooperative State University Loerrach, in 1999. In 2011, he joined Offenburg University of

Applied Sciences, where he leads the Institute of Reliable Embedded Systems and Communication Electronics. Since 2016, he is also deputy member of the board at Hahn-Schickard Association of Applied Research, where he heads the "Software Solutions" division. His major interests include field of efficient, energy-aware, safe, and secure algorithms and protocols for wired and wireless embedded communication. In 2002, he founded the Steinbeis Transfer Center Embedded Design and Networking for professional protocol and platform developments, which was successfully spun off as STACKFORCE GmbH in 2014. He is the author, co-author, editor, and co-editor of several textbooks and numerous papers in the field of embedded design and wireless and wired networking, and head and member of manifold steering and program committees of international scientific conferences.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Group Signatures with Time-Bound Keys Revisited: A New Model, an Efficient Construction, and its Implementation

Keita Emura , Takuya Hayashi, and Ai Ishida

Abstract—Chu et al. (ASIACCS 2012) proposed group signature with time-bound keys (GS-TBK), where each signing key is associated with expiry time τ . In addition, to prove membership of the group, a signer needs to prove that the expiry time has not passed, i.e., $t < \tau$, where t is the current time. A signer whose expiry time has passed is automatically revoked, and this revocation is called natural revocation. Signers can be revoked simultaneously before their expiry times if the credential is compromised. This revocation is called premature revocation. A nice property in the Chu et al. proposal is that the size of revocation lists can be reduced compared to those of Verifier-Local Revocation (VLR) group signature schemes by assuming that natural revocation accounts for most of the signer revocations in practice, and prematurely revoked signers are only a small fraction. In this paper, we point out that the definition of traceability of Chu et al. did not capture the unforgeability of expiry time for signing keys, which guarantees that no adversary who has a signing key associated with expiry time τ can compute a valid signature after τ has passed. This situation significantly reduces the dependability of the system since legitimate signing keys may be used for providing a forged signature. We introduce a security model that captures unforgeability, and propose a secure GS-TBK scheme in the new model. Our scheme also provides constant signing costs, whereas those of the previous schemes depended on the bit-length of the time representation. Finally, we provide the implementation results. We employ Barreto-Lynn-Scott (BLS) curves with 455-bit prime order and the RELIC library, and demonstrate that our scheme is feasible in practical settings.

Index Terms—Group signatures, time-bound keys, revocation

1 INTRODUCTION

1.1 Group Signatures and Revocation

GROUP signatures, proposed by Chaum and van Heyst [2], provide functionality to anonymously prove the membership of a group. After the seminal work by Boneh, Boyen, and Shacham (BBS) [3], many pairing-based constructions have been proposed, e.g., [4], [5], [6], [7], [8], [9], [10], [11], [12]. Recently, lattice-based constructions have also been proposed, e.g., [13], [14]. Among them, providing revocation functionality¹ is regarded as a major research topic for group signatures, where an authority can revoke the membership of users. One reason for the difficulty in providing revocation functionality in the context of group signatures is that a

verifier needs to publicly confirm whether an anonymous signer has been revoked or not. To overcome this difficulty, several attempts have been made so far.

In revocable group signatures, there are two checks in the verification algorithm, the verification check and the revocation check. After that, almost all currently known revocable group signature schemes can be classified as follows.

- (1) Revoked signers cannot compute a signature that passes the verification check (and therefore no revocation check procedure is required for this type) [15], [16], [17], [18], [19], [20], [21], [22], [23].
- (2) Any signer can compute a signature that passes the verification check, but a verifier can check whether the signer has been revoked or not using the revocation check procedure [7], [24], [25], [26], [27], [28], [29].

An example of the first type of scheme is the Libert-Peters-Yung revocable group signature scheme [18]: a ciphertext of broadcast encryption is published such that non-revoked signers are regarded as authorized users and they can decrypt the ciphertext. A non-revoked signer proves the decryption ability of the ciphertext as evidence that the signer is not revoked. Since revoked signers cannot decrypt the ciphertext, revoked signers cannot compute a signature that passes the verification check in the first place. For this type, a signer not only needs to prove membership of the group, but also that the signer has not been revoked, and the computational cost of the signing algorithm is relatively high compared to that of the second type scheme. On the other hand, the computational cost of the

1. We clearly distinguish revocation and anonymity revocation. The former means that signing keys are expired whereas the latter means that an authority called opener identifies the signer.

- K. Emura is with National Institute of Information and Communications Technology (NICT), 4-2-1, Nukui-kitamachi, Koganei, Tokyo 184-8795, Japan. E-mail: k-emura@nict.go.jp.
- T. Hayashi is with Kobe University, and National Institute of Information and Communications Technology (NICT), 1-1, Rokko-dai, Nada-ku, Kobe 657-8501, Japan. E-mail: t-hayashi@eedept.kobe-u.ac.jp.
- A. Ishida is with Tokyo Institute of Technology, and National Institute of Advanced Industrial Science and Technology (AIST), 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan. E-mail: ishida0@is.titech.ac.jp.

Manuscript received 27 June 2017; revised 5 Sept. 2017; accepted 14 Sept. 2017. Date of publication 19 Sept. 2017; date of current version 18 Mar. 2020. (Corresponding author: Keita Emura.)

Digital Object Identifier no. 10.1109/TDSC.2017.2754247

verification algorithm is relatively low compared to that of the first type of scheme. More precisely, the verification cost is constant in terms of the number of revoked signers.

An example of the second type of scheme is the Boneh-Shacham Verifier-Local Revocation group signature scheme [24]: a signer is not involved in the revocation procedure. Thus, any signer can compute a signature that passes the verification check. In order to check whether the signer of the signature is revoked or not, the verifier runs the revocation check procedure using a revocation list RL_t that contains information about the revoked signers at time period t . Since a signer only needs to prove membership of the group, the computational cost of the signing algorithm is relatively low compared to that of the first type of scheme. On the other hand, the computational cost of the verification algorithm is relatively high compared to that of the first type of scheme, due to the additional revocation check procedure. Usually, the computational cost of the revocation check grows linearly depending on the size of the revocation list. Thus, reducing the size of the revocation list is highly desirable. However, information about revoked signers is added to the revocation list each time, and its size grows periodically.

1.2 Group Signatures with Time-Bound Keys

Chu et al. [30] proposed group signature with time-bound keys (GS-TBK). We can regard that GS-TBK has the simultaneous properties of both revocation types. Expiry time τ is set for each signing key, and in addition to prove the membership to the group, a signer needs to prove that the expiry time has not passed, i.e., $t < \tau$ where t is the current time. Since signers whose expiry time has passed cannot compute a signature that passes the verification check in the first place, this can be classified as the first type. Chu et al. called this “natural” revocation. Simultaneously, signers can be revoked before their expiry times have passed due to a compromised credential. Since a verifier runs the revocation check procedure, this can be classified as the second type. Chu et al. called this “premature” revocation.

One nice property of the Chu et al. proposal is that the size of a revocation list can be reduced compared to those in VLR group signature schemes, by assuming that natural revocation accounts for most of the signer revocations in practice, and prematurely revoked signers are only a small fraction. That is, a revocation list RL_t just needs to contain information about revoked signers whose expiry time τ has not passed (i.e., $t < \tau$). A small revocation list leads to the reduction of the costs of revocation check.

1.3 Our Target and Contribution: A New Model and an Efficient Construction

We point out that the definition of traceability of Chu et al. [30] (and its journal version [31]) did not capture the following case:

- *Forgery of expiry time*: An adversary who has a signing key associated with expiry time τ may compute a valid signature after τ has passed.

More precisely, the winning condition of the adversary of traceability in [30], [31] is defined as follows. Let (σ^*, m^*) be the output of the adversary and t^* be the time that the adversary outputs (σ^*, m^*) . Then, it is required that

- (1) σ^* is a valid signature on message m^* with revocation list RL_{t^*} .
- (2) σ^* is not obtained from the signing queries with t^* on m^* .
- (3) σ^* is NOT traced to a signer in $CU \setminus RU_{t^*}$ or the trace has failed, where CU is the set of corrupted signers, i.e., the adversary has their signing keys, and RU_{t^*} is a set of revoked signers at t^* .

It seems natural to additionally consider the case that

- (4) σ^* is traced to a signer in $CU \setminus RU_{t^*}$ and $\tau^* < t^*$ holds where τ^* is the expiry time of the corresponding signing key of the traced signer.

This unforgeability of expiry time should be considered due to the usage of time-bound signing keys. We remark that we have not found any particular attack against the schemes [30], [31] in the new model. Nevertheless, it seems meaningful to provide a provably secure scheme in the new model. In addition to the unforgeability of expiry time, we also consider backward unlinkability [27] and non-frameability under the dishonest group manager setting, which were not considered in [30], [31].

The next target is efficiency, since the signing cost and the signature size of the Chu et al. scheme [30] linearly depend on $\log T$ where T is the maximum-length of time t . They apply the encoding technique proposed by Lin and Tzeng [32] for proving $t < \tau$. In the journal version [31], by using accumulators [22] together with the encoding, the signature size can be constant whereas the signing cost still linearly depends on $\log T$.

Our Contribution: In this paper, we define a new model of GS-TBK that captures the unforgeability of expiry time for signing keys, and propose a secure GS-TBK scheme with this model. Moreover, in our scheme, the cost of the signing algorithm is constant, whereas those of the previous schemes [30], [31] depend on the bit-length of the time representation. In addition, we further reduce the size of the revocation list compared to previous ones. We give the efficiency comparison in Table 1.

For proving that the expiry time has not passed, we employ the Ohara et al. methodology [21], which efficiently implements the Libert-Peters-Yung revocable group signature scheme [18] in the random oracle model. Ohara et al. employed the Complete Subtree (CS) method [34] for revocation, as in the Libert-Peters-Yung (CS-based) construction, where each signer is assigned to the leaf node of a tree structure. The Libert-Peters-Yung scheme uses an identity-based encryption (IBE) scheme for instantiating the CS method to the public key setting [35]. Ohara et al. used a signature scheme instead of an IBE scheme. A revocation list contains the signatures of nodes, which are determined by the CS method. In other words, signatures for revoked signers are not contained in the list. Non-revoked signers can prove that a signature related to the signers is contained in the list. We employ the Ohara et al. methodology such that time t and expiry time τ are assigned to leaves in a sequential order, and at time t , leaves associated with $1, 2, \dots, t-1$ are revoked. For the natural revocation, the group manager broadcasts expiry information ei_t . If $t < \tau$, then a signer whose signing key is not expired can prove that τ has not been revoked using ei_t . This ei_t helps signers to efficiently

TABLE 1
Efficiency Comparison of Group Signature Schemes with Time-Bound Keys

Scheme	Group public key size	Signature size	Signing key size	Revocation list size	Expiration info size	Signing cost	Verification cost	BU/ UET	Model	KOSK Assump.
Chu et al. [30]	$O(1)$	$O(\log T)$	$O(\log T)$	$O(R_{\text{pre}} \log T)$	-	$O(\log T)$	$O(R_{\text{pre}})^1$	NO	ROM	-
Liu et al. [31]	$O(\log T)$	$O(1)$	$O(\log T)$	$O(R_{\text{pre}} \log T)$	-	$O(\log T)$	$O(R_{\text{pre}})^1$	NO	ROM	-
Ours	$O(1)$	$O(1)$	$O(\log T)$	$O(R_{\text{pre}})$	$O(\log T)$	$O(1)$	$O(R_{\text{pre}})$	YES	ROM	Require

T : The maximum size of expiry time.

R_{pre} : The number of “prematurely” revoked signers.

BU: Backward Unlinkability

UET: Unforgeability of Expiry Time for Signing Keys

ROM: Random Oracle Model

KOSK: Knowledge of Secret Key [33]

¹More precisely, this complexity is represented as $O(\log T + R_{\text{pre}})$. Here, we assume that $\log T < R_{\text{pre}}$.

prove that expiry time τ has not passed from the current time t where $t < \tau$ without showing τ . One drawback in our construction is that signers need to download expiration information ei_t at each time t although neither an encryption for sending ei_t nor updating the secret key is required. In the meantime, no additional expiry time-related value for signing is required in the previous schemes [30], [31]. At this expense, we can achieve an $O(1)$ signing cost and $O(R_{\text{pre}})$ -size revocation list, whereas those of the previous schemes are $O(\log T)$ and $O(R_{\text{pre}} \log T)$ respectively, where T is the maximum size of the expiry time and R_{pre} is the number of “prematurely” revoked signers. We remark that in our scheme signers are still not required to obtain signer revocation-related information, i.e., a revocation list, for generating signatures. Moreover, ei_t is independent from the premature revocation, and its size does not depend on the number of revoked signers. As other drawback of our scheme, from the technical reason, we need to require an additional assumption, the knowledge of secret key (KOSK) assumption [33] as in the Ohara et al. scheme, where a new user is required to prove the knowledge of the secret key corresponding its public key. In the security proofs, the adversary is required to directly reveal the secret key of the honest users. Usually, the KOSK assumption is required to prevent rogue-key attacks where adversaries can choose their public key arbitrary. Our scheme and also the Ohara et al. scheme require the KOSK assumption in order to send a signing query to the underlying signature scheme without knowing the message. See Section 4 for details. Though the KOSK assumption is employed in some works, e.g., [36], [37], there are substantial drawbacks of the KOSK assumption as discussed in [33]. For example, the KOSK assumption is not realized by existing public key infrastructures (PKI). Thus, it would be better to avoid to employ the KOSK assumption as much as possible, and removing the assumption is stated as a future work of this paper.

Finally, we implement our scheme and demonstrate that it provides enough efficiency in practice. In our implementation we employ Barreto-Lynn-Scott curves [38] of embedding degree 12 over a 455-bit prime field to ensure almost 128-bit security [39], [40], and use the RELIC library [41].

We expect that we can efficiently add anonymity to existing authentication systems based on group signature schemes since our revocable group signature scheme yields efficient signing/verification algorithm, and provides a smaller-size revocation list. For example, several attempts

for adding anonymity to the standard X.509 certificates have already been made, e.g., [42], [43], [44], and in addition to them, revocation in X.509 with anonymity has also been investigated, e.g., [45], [46]. We remark that the method in [45] requires traceable signature [47], [48], and it seems not straightforward to apply our scheme to them.

1.4 Related Work

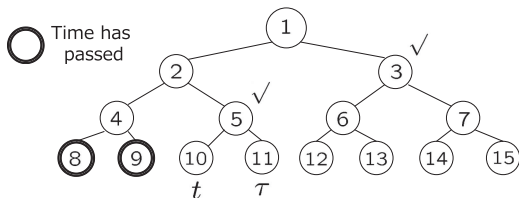
Malina et al. [49], [50] also proposed group signatures with time-bound membership. However, different from ours and Chu et al. [30], [31], they do not consider premature revocation. Moreover, some information of expiry time (index k in [49], [50]) has to be contained in a signature. As mentioned in [30], [31], signers may not wish to leak such information (even partially) because it may be used to infer signers’ identity. Hence, as in [30], [31], a signer is allowed to completely (i.e., in the sense of zero-knowledge proofs) hide his/her expiry time in our scheme.

The first VLR group signature scheme was proposed by Boneh and Shacham [24], and Nakanishi and Funabiki [27] considered the notion called backward unlinkability. A signature contains a target group (i.e., \mathbb{G}_T) element in their scheme. Later, they proposed a more efficient VLR group signature scheme [28] whose signature contains base group (or \mathbb{Z}_p) elements only. Hence, we employ the Nakanishi-Funabiki scheme proposed in [28] with a slight modification due to the curve selection since they employed (Type-2) MNT curves [51] whereas we employ (Type-3) BLS curves.

1.5 Differences from the Proceedings Version

Here, we explain the additional contents from the proceedings version [1]. We mainly reconsider two parts: the security parameters and the pairing equations in our algorithms, and re-implement the proposed scheme.

To implement our GS-TBK scheme, we use the RELIC library [41] for elliptic curve operations and the pairing operation. In the proceedings version [1], we chose a Barreto-Naehrig (BN) curve over a 254-bit prime field, which was believed to have 128-bit security. However, recent progress in discrete logarithm problems of finite fields [52], [53] asymptotically reduces the complexity of the problems, and recent evaluations [39], [40] show that the 254-bit BN curve cannot ensure 128-bit security. Based on Barbulescu-Duquesne’s estimate [40], we choose a Barreto-Lynn-Scott curve [38] with an embedding degree of 12 over a 455-bit prime field to ensure almost 128-bit security, which is supported by the RELIC library.

Fig. 1. τ has not passed.

Increasing the size of base field affects the performance of our scheme. Therefore we reconsider the computation of our algorithms to reduce their costs. Let $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T be pairing groups of order p . The cost of scalar multiplication on $\mathbb{G}_1, \mathbb{G}_2$, and exponentiation on \mathbb{G}_T is different, and exponentiation on \mathbb{G}_T is generally slower than scalar multiplication on the other groups. Moreover, it is known that scalar multiplication/exponentiation can be accelerated when a base point is previously known and fixed. We measure operation benchmarks on these groups and modify the computation of our algorithms using bilinearity of pairing to optimize the performance of the algorithms for the 455-bit BLS curve of the RELIC library. Our optimization allows us to reduce the running time of the signing algorithm by approximately 45 percent, and also to reduce the running time of the verification check by approximately 50 percent. We provide the algorithm modification and the implementation result in Section 5.

2 PRELIMINARIES

In this section, we define the Complete Subtree algorithm for time-bound keys (CS-TBK), complexity assumptions, and the BBS+ signature scheme [3], [54]. First, we give the definition of the CS-TBK algorithm which implements (a special case of) the CS method [34]. Let BT be a binary tree that has T leaf nodes where T is the maximum size of time. The algorithm finds subtrees that cover all non-revoked nodes. Note that, in the Ohara et al. revocable group signature scheme, each user is assigned to a leaf whereas each time is sequentially assigned to a leaf node in the algorithm.

Definition 2.1 (The CS-TBK Algorithm). *This algorithm takes as input a binary tree BT and the current time t , and outputs a set of nodes. If η is a non-leaf node, then η_{left} and η_{right} denote the left and right child of η , respectively. Each time is sequentially assigned to a leaf node. $\text{Path}(\eta)$ denotes the set of nodes on the path from η to the root node. The description of the algorithm is given below.*

CS-TBK(BT, t) :

$X, Y \leftarrow \emptyset$;

$\forall 1 \leq i < t$

 Add $\text{Path}(\eta)$ to X where i is assigned to η

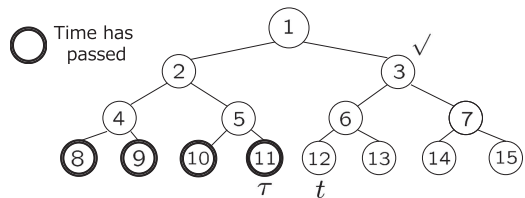
$\forall x \in X$

 If $x_{\text{left}} \notin X$ then add x_{left} to Y

 If $x_{\text{right}} \notin X$ then add x_{right} to Y

 If $Y = \emptyset$ then add root to Y

Return Y

Fig. 2. τ has passed.

In our GS-TBK scheme, each time t is assigned to a leaf node, and expiry time τ is also assigned to a leaf node. That is, one leaf node is shared by multi signers if their expiry times are the same. If τ is assigned to a leaf node η , the group manager generates signatures of nodes contained in $\text{Path}(\eta)$, and then these signatures are issued to signers whose expiry time is τ . We remark that the randomness of these signatures is different for signers even they share the same leaf node. At current time t , all leaf nodes of past time, i.e., all left-side leaves of the leaf node assigned to t are revoked.² Next, the group manager generates signatures of nodes generated by the CS-TBK algorithm, and publishes signatures as expiration information ei_t at time t . If $t < \tau$, then the corresponding signers have a signature of a node such that ei_t contains the signature of the same node.

We give an example in the case of $T = 8$ as follows. We show a case that τ has not passed in Fig. 1, and also show a case that τ has passed in Fig. 2. Let τ be assigned to the node 11. Then, signers whose expiry time is τ have signatures of nodes 1, 2, 5, and 11. In Fig. 1, nodes 8 and 9 are revoked. Then, nodes 3 and 5 are selected as roots of subtrees. Thus, ei_t contains signatures of nodes 3 and 5. Then, the signers can prove that they have a signature of the node 5 (without revealing the node itself). In Fig. 2, ei_t contains a signature of the node 3 only. Since τ has passed, signatures of 1, 2, 5, and 11 are not contained in ei_t .

Next, we define complexity assumptions. Let p be a λ -bit prime, $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are groups of order p , $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, and g, \hat{g} are generators of \mathbb{G}_1 and \mathbb{G}_2 , respectively. We use the asymmetric setting (Type-3 curves), i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$, and no efficient isomorphism between \mathbb{G}_1 and \mathbb{G}_2 is known.

Next, we define decision Diffie-Hellman assumption on \mathbb{G}_1 (DDH1) as follows.

Definition 2.2 (DDH1 Assumption). *Let $D := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g})$, $a, b \xleftarrow{\$} \mathbb{Z}_p$ and $Z \xleftarrow{\$} \mathbb{G}_1 \setminus \{g^{ab}\}$. We say that the DDH1 assumption holds if for any probabilistic polynomial time (PPT) adversary \mathcal{A} , the advantage $\text{Adv}_{\text{DDH1}}(\lambda) := |\Pr[\mathcal{A}(D, g^a, g^b, g^{ab}) \rightarrow \text{true}] - \Pr[\mathcal{A}(D, g^a, g^b, Z) \rightarrow \text{true}]|$ is negligible.*

Next, we define the decision linear (DLIN) assumption. Here, we use an asymmetric variant [56].

Definition 2.3 (DLIN Assumption). *Let $D := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \tilde{g}, \hat{g})$, $a, b, c, d \xleftarrow{\$} \mathbb{Z}_p$, $g' := \tilde{g}^c$, $\tilde{g}' := \hat{g}^c$, $h := \tilde{g}^d$, $\hat{h} := \hat{g}^d$, and*

2. In the usual CS method, a user is associated to a leaf node, and who will be revoked is not predictable. So, the size of Y is $O(r \log(N/r))$ where N is the number of users (leaves), and r is the number of revoked signers. On the other hand, in our usage, though a time is associated to a leaf node as usual, leaves are “sequentially” revoked. So, the size of Y is at most $\log T$. This is essentially the same as the encoding for attribute-based encryption with range membership [55].

$Z \stackrel{\$}{\leftarrow} \mathbb{G}_1 \setminus \{h^{a+b}\}$. We say that the DLIN assumption holds if for any PPT adversary \mathcal{A} , the advantage $\text{Adv}_{\text{DLIN}}(\lambda) := |\Pr[\mathcal{A}(D, g', \hat{g}', h, \hat{h}, \hat{g}^a, g^b, h^{a+b}) \rightarrow \text{true}] - \Pr[\mathcal{A}(D, g', \hat{g}', h, \hat{h}, \hat{g}^a, g^b, Z) \rightarrow \text{true}]|$ is negligible.

Next, we define the q -Strong Diffie-Hellman (q -SDH) assumption as follows. Here, we use an asymmetric variant [57].

Definition 2.4 (q -SDH Assumption). Let $D := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g})$ and $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. We say that the q -SDH assumption holds if for any PPT adversary \mathcal{A} , the advantage $\text{Adv}_{q\text{-SDH}}(\lambda) := \Pr[\mathcal{A}(D, g^x, g^{x^2}, \dots, g^{x^q}, \hat{g}^x) \rightarrow (c, g^{1/(x+c)}) \in \mathbb{Z}_p \setminus \{-x\} \times \mathbb{G}_1]$ is negligible.

Next, we define the discrete logarithm (DL) assumption (on \mathbb{G}_1) as follows.

Definition 2.5 (DL Assumption). Let $D := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g})$ and $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. We say that the DL assumption holds if for any PPT adversary \mathcal{A} , the advantage $\text{Adv}_{\text{DL}}(\lambda) := \Pr[\mathcal{A}(D, g^x) \rightarrow x]$ is negligible.

Next, we introduce the BBS+ signature scheme [3], [54], especially, the BBS+ signature scheme over a Type-3 curve [57]. This scheme allows to sign L messages, and is existential unforgeable against chosen message attack under the q -SDH assumption. Let g, h_0, h_1, \dots, h_L be generators of \mathbb{G}_1 , \hat{g} be a generator of \mathbb{G}_2 , and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map. The BBS+ signature scheme [3], [54]

- **Key Generation:** Choose $\gamma \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, and let $w = \hat{g}^\gamma$. The verification key is $vk = w$, and the secret key is $sk = \gamma$.
- **Sign:** For the messages $(m_1, \dots, m_L) \in \mathbb{Z}_p^L$, choose $\xi, \zeta \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and compute $A = (gh_0^\xi h_1^{m_1} \dots h_L^{m_L})^{\frac{1}{\xi+\gamma}}$. Output the signature $\sigma = (A, \xi, \zeta)$.
- **Verify:** For a signature $\sigma = (A, \xi, \zeta)$ and messages (m_1, \dots, m_L) , if $e(A, \hat{g}^\gamma vk) = e(gh_0^\xi h_1^{m_1} \dots h_L^{m_L}, \hat{g})$ holds, then output 1, and otherwise output 0.

3 DEFINITION OF GROUP SIGNATURES WITH TIME-BOUND KEYS

In this section, we give the definition of GS-TBK, mainly following the definition of Chu et al. [30], [31]. We additionally introduce the unforgeability of expiry time for signing keys, backward unlinkability, and non-frameability against a malicious group manager. Moreover, our model introduces expiration information ei_t .

In contrast to the model of group signatures [58], [59], [60], [61], a revocation token grt_i is generated when signer i joins the group. Revocation tokens are modified according to current time period t . We denote it as $grt_{i,t}$, and $grt_{i,t}$ is contained in the revocation list RL_t if signer i is revoked at t , and is used for the revocation check. We emphasize that if $\tau_i < t$, then $grt_{i,t}$ does not have to be contained in RL_t since the expiry time has passed. On the other hand, in the case of VLR group signatures, all $grt_{i,t}, \dots, grt_{i,T}$ need to be contained in RL_t . So, the size of revocation list can be reduced due to time-bound keys. Moreover, in the model, signing key gsk_i is associated with expiry time τ_i , and the signing algorithm takes current time t as an input.

A GS-TBK scheme $\mathcal{GS}\text{-TBK}$ consists of six algorithms: (GKeyGen, Join/Issue, Revoke, Sign, Verify, Open) which are defined as follows.

Definition 3.1 (Syntax of GS-TBK).

- **GKeyGen:** The group key generation algorithm takes as input a security parameter $\lambda \in \mathbb{N}$, and outputs a group public key gpk and a master secret key msk . Set a registration table $\text{reg} := \emptyset$. We assume that the maximum size of expiry time T also contained in gpk .
- **Join/Issue:** This is the pair of interactive algorithms which implement the joining protocol run by a user i and the group manager. The joining algorithm Join takes as input gpk , whereas the issuing algorithm Issue takes as input msk , reg , and an expiry time τ_i . Upon successful completion of the protocol, the Join algorithm outputs a signing key gsk_i , τ_i , and a user secret key usk_i , and the Issue algorithm outputs reg where $\text{reg}[i]$ stores a revocation token grt_i and τ_i .
- **Revoke:** The revocation algorithm takes as input gpk , msk , t , reg , and a set of signers to be revoked at t RU_t . Let a set of their revocation tokens grt_i and its expiry time τ_i be $\{\text{reg}[i] := (grt_i, \tau_i)\}$ which are contained in reg . Set $\text{RL}_t := \emptyset$. For each i , the algorithm computes $grt_{i,t}$ if τ_i has not passed, i.e., $t < \tau_i$, and stores $grt_{i,t}$ to RL_t . Moreover, the algorithm computes expiration information ei_t . Finally, the algorithm outputs (ei_t, RL_t) .
- **Sign:** The signing algorithm takes as input gpk , gsk_i , usk_i , a message to be signed m , the current time t , and ei_t , and outputs a signature σ .
- **Verify:** The verification algorithm takes as input gpk , t , σ , m , and RL_t , and outputs either valid or invalid.
- **Open:** The opening algorithm takes as input gpk , msk , t , reg , σ , m , and RL_t , and outputs the identity of the signer i or \perp .

The correctness is defined as follows. This guarantees that a group signature generated by a signing key with τ_i at time t^* , where $t^* < \tau_i$ and the signer i is not revoked at time t^* , is valid, and the opening result correctly indicates i .

Definition 3.2 (Correctness). For any PPT adversary \mathcal{A} and the security parameter $\lambda \in \mathbb{N}$, we define the experiment $\text{Exp}_{\text{GS-TBK}, \mathcal{A}}^{\text{corr}}(\lambda)$ as follows.

$\text{Exp}_{\text{GS-TBK}, \mathcal{A}}^{\text{corr}}(\lambda) :$

$(gpk, msk, \text{reg}) \leftarrow \text{GKeyGen}(\lambda); \text{HU} := \emptyset$

$(i, m, t^*) \leftarrow \mathcal{A}^{\text{AddU, RReg, Revoke}}(gpk); i \in \text{HU} \setminus \text{RU}_{t^*}; t^* < \tau_i$

$\sigma \leftarrow \text{Sign}(gpk, gsk_i, usk_i, m, t^*, ei_{t^*})$

$j \leftarrow \text{Open}(gpk, msk, t^*, \text{reg}, \sigma, m, \text{RL}_t)$

Return 1 if the following holds :

$\text{Verify}(gpk, t^*, \sigma, m, \text{RL}_{t^*}) = \text{invalid} \vee i \neq j$

Otherwise return 0.

- **AddU:** The add user oracle allows an adversary \mathcal{A} to add honest users to the group. On input an identity i and τ_i , this oracle computes (gsk_i, τ_i) by running Join/Issue. i is added to HU.
- **RReg:** On input i , the read-registration-table oracle reveals the content of the registration table $\text{reg}[i]$.

- **Revoke:** Let $t - 1$ be the time that the oracle is called. The revocation oracle allows \mathcal{A} to revoke honest users. On input identities RU_t , this oracle runs $\text{RL}_t \leftarrow \text{Revoke}(gpk, msk, t, \text{reg}, \text{RU}_t)$, and outputs (ei_t, RL_t) . We say that $\mathcal{GS}\text{-TBK}$ is correct if the advantage

$$\text{Adv}_{\mathcal{GS}, \mathcal{A}}^{\text{corr}}(\lambda) := \Pr[\text{Exp}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{corr}}(\lambda) = 1],$$

is negligible for any PPT adversary \mathcal{A} .

The anonymity with backward unlinkability (BU-anonymity) is defined as follows. This guarantees that no signer identity is revealed from signatures even the corresponding signer has been revoked. We follow selfless CPA anonymity [30], [31] where an adversary is allowed to obtain signing keys except the challenge users' keys,³ and is not allowed to access the open oracle.

Definition 3.3 (BU-Anonymity). For any PPT adversary \mathcal{A} and a security parameter $\lambda \in \mathbb{N}$, we define the experiment $\text{Exp}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{bu-anon}}(\lambda)$ as follows.

$\text{Exp}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{bu-anon}}(\lambda) :$
 $b \xleftarrow{\$} \{0, 1\}$
 $(gpk, msk, \text{reg}) \leftarrow \text{GKeyGen}(\lambda)$
 $\text{HU} := \emptyset; \text{CU} := \emptyset; \text{RU} := \emptyset$
 $b' \leftarrow \mathcal{A}^{\text{AddU, WReg, USK, Revoke, GSign, Ch}_b}(gpk)$
 Return 1 if $b' = b$, and 0 otherwise.

- **AddU:** The add user oracle allows an adversary \mathcal{A} to add honest users to the group. On input an identity i and τ_i , this oracle computes (gsk_i, τ_i) by running Join/Issue. i is added to HU .
- **WReg:** On input i and M , the write-registration-table oracle updates $\text{reg}[i]$ to M .
- **USK:** On input i , the user-secret-keys oracle reveals (gsk_i, usk_i) and adds i to CU .
- **Revoke:** Let $t - 1$ be the time that the oracle is called. The revocation oracle allows \mathcal{A} to revoke honest users. On input identities RU_t , this oracle runs $\text{RL}_t \leftarrow \text{Revoke}(gpk, msk, t, \text{reg}, \text{RU}_t)$, outputs (ei_t, RL_t) , adds RU_t to RU . Remark that i_0 and i_1 can be revoked if $t^* < t$.
- **GSign:** On input i and m where $i \in \text{HU}$, the signing oracle computes $\sigma \leftarrow \text{Sign}(gpk, gsk_i, usk_i, m, t, ei_t)$ and returns σ . Here, t is the current time that the oracle called.
- **Ch_b:** On input i_0, i_1 , where $i_0, i_1 \in \text{HU}$, and m^* , the challenge oracle computes $\sigma^* \leftarrow \text{Sign}(gpk, gsk_{i_b}, usk_{i_b}, m^*, t^*, ei_{t^*})$ and returns σ^* . Here, $i_0, i_1 \notin \text{CU}$, $i_0, i_1 \notin \text{RU}$, $t^* < \tau_{i_0}$, and $t^* < \tau_{i_1}$ must hold.

3. We call anonymity full anonymity if the adversary is allowed to obtain all signing keys. As a theoretical result, selfless anonymity is weaker than full anonymity since the former can be constructed from one-way functions and non-interactive zero-knowledge (NIZK) arguments [62] whereas the latter implies public key encryption [63], [64], [65]. We employ the selfless anonymity in this paper, as in the previous works [30], [31] and VLR group signature schemes since a revocation token can be computed by a signing key.

We say that $\mathcal{GS}\text{-TBK}$ is BU-anonymous if the advantage

$$\text{Adv}_{\mathcal{GS}\text{-TDL}, \mathcal{A}}^{\text{bu-anon}}(\lambda) := |\Pr[\text{Exp}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{bu-anon}}(\lambda) = 1] - 1/2|,$$

is negligible for any PPT adversary \mathcal{A} .

The traceability is defined as follows. We mainly follow the definition of [30], [31] except that we additionally consider the unforgeability of expiry time for signing keys (the winning condition (4) in the experiment). Traceability guarantees that no adversary who does not have a signing key can compute a valid signature. Moreover, it guarantees that a valid signature can be traced, and no adversary can produce a valid signature using a secret key after the expiry time of the signing key has passed.

Definition 3.4 (Traceability). For any PPT adversary \mathcal{A} and a security parameter $\lambda \in \mathbb{N}$, we define the experiment $\text{Exp}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{trace}}(\lambda)$ as follows.

$\text{Exp}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{trace}}(\lambda) :$
 $(gpk, msk, \text{reg}) \leftarrow \text{GKeyGen}(\lambda); \text{CU} := \emptyset; \text{SSet} := \emptyset$
 $(\sigma^*, m^*, t^*) \leftarrow \mathcal{A}^{\text{SndTol, RReg, Revoke, GSign}}(gpk)$
 Return 1 if (1) \wedge (2) \wedge ((3) \vee (4)) holds :
 (1) $\text{Verify}(gpk, t^*, \sigma^*, m^*, \text{RL}_{t^*}) = \text{valid}$
 (2) $(t^*, \sigma^*, m^*) \notin \text{SSet}$
 (3) $i \leftarrow \text{Open}(gpk, msk, t^*, \text{reg}, \sigma^*, m^*, \text{RL}_{t^*})$
 $\wedge (i \notin \text{CU} \setminus \text{RU}_{t^*} \vee i = \perp)$
 (4) $i \leftarrow \text{Open}(gpk, msk, t^*, \text{reg}, \sigma^*, m^*, \text{RL}_{t^*})$
 $\wedge i \in \text{CU} \setminus \text{RU}_{t^*} \wedge \tau_i < t^*$
 Otherwise return 0.

- **SndTol:** The send-to-issuer oracle allows \mathcal{A} to engage the joining protocol on behalf of the corrupted user i . Finally, gsk_i , τ_i , and usk_i are given to \mathcal{A} , and i is added to CU .
- **RReg:** On input i , the read-registration-table oracle reveals the content of the registration table $\text{reg}[i]$.
- **Revoke:** Let $t - 1$ be the time that the oracle is called. The revocation oracle allows \mathcal{A} to revoke honest users. On input identities RU_t , this oracle runs $\text{RL}_t \leftarrow \text{Revoke}(gpk, msk, t, \text{reg}, \text{RU}_t)$, and outputs (ei_t, RL_t) .
- **GSign:** On input i and m , the signing oracle computes $\sigma \leftarrow \text{Sign}(gpk, gsk_i, usk_i, m, t, ei_t)$, returns σ , and adds (t, σ, m) to SSet . Here, t is the current time that the oracle called.

We say that $\mathcal{GS}\text{-TBK}$ is traceable if the advantage

$$\text{Adv}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{trace}}(\lambda) := \Pr[\text{Exp}_{\mathcal{GS}\text{-TBK}, \mathcal{A}}^{\text{trace}}(\lambda) = 1],$$

is negligible for any PPT adversary \mathcal{A} .

The non-frameability is defined as follows. This guarantees that no adversary can produce a valid signature which is traced to an honest signer. Our definition allows that the group manager is corrupted, i.e., an adversary is given msk and is allowed to read reg . Here, honest means that the signer (i^* in the experiment) is added to the group via the SndToU oracle, and the USK oracle for i^* is not called. That is, the adversary does not know usk_{i^*} .

Definition 3.5 (Non-frameability). For any PPT adversary \mathcal{A} and a security parameter $\lambda \in \mathbb{N}$, we define the experiment $\text{Exp}_{\text{GS-TBK},\mathcal{A}}^{\text{nf}}(\lambda)$ as follows.

$\text{Exp}_{\text{GS-TBK},\mathcal{A}}^{\text{nf}}(\lambda) :$
 $(gpk, msk, \text{reg}) \leftarrow \text{GKeyGen}(\lambda)$
 $\text{HU} := \emptyset; \text{CU} := \emptyset; \text{SSet} := \emptyset$
 $(\sigma^*, m^*, t^*, i^*) \leftarrow \mathcal{A}^{\text{SndToU, RReg, USK, GSign}}(gpk, msk)$
 Return 1 if the following holds :
 (1) $\text{Verify}(gpk, t^*, \sigma^*, m^*, \text{RL}_{t^*}) = \text{valid}$
 (2) $i^* \leftarrow \text{Open}(gpk, msk, t^*, \text{reg}, \sigma^*, m^*, \text{RL}_{t^*})$
 (3) $i^* \in \text{HU} \wedge i^* \notin \text{CU} \wedge (t^*, \sigma^*, m^*) \notin \text{SSet}$
 Otherwise return 0.

- **SndToU:** The send-to-user oracle allows \mathcal{A} to engage a joining protocol of the user i on the behalf of the corrupted group manager. i is added to HU .
- **RReg:** On input i , the read-registration-table oracle reveals the content of the registration table $\text{reg}[i]$.
- **USK:** On input i , the user-secret-keys oracle reveals (gsk_i, usk_i) and adds i to CU .
- **GSign:** On input i and m , the signing oracle computes $\sigma \leftarrow \text{Sign}(gpk, gsk_i, usk_i, m, t, ei_t)$, returns σ , and adds (t, σ, m) to SSet .

Here, t is the current time that the oracle called. We say that GS-TBK is non-frameable if the advantage

$$\text{Adv}_{\text{GS-TBK},\mathcal{A}}^{\text{nf}}(\lambda) := \Pr[\text{Exp}_{\text{GS-TBK},\mathcal{A}}^{\text{nf}}(\lambda) = 1],$$

is negligible for any PPT adversary \mathcal{A} .

4 THE PROPOSED GS-TBK SCHEME

In this section, we give the proposed GS-TBK scheme. For the natural revocation, we employ the Ohara et al. revocable group signature scheme [21], and for the premature revocation, we employ the Nakanishi-Funabiki VLR group signature scheme [28]. We slightly modify the Nakanishi-Funabiki scheme since our scheme is constructed over a Type-3 curve, whereas the Nakanishi-Funabiki scheme is constructed over a Type-2 curve. As mentioned in [27], [28], revocation tokens can be implicitly used for tracing signers. That is, the group manager computes $grt_{i,t}$ for all grt_i , and checks the revocation check equation. If the equation holds for $grt_{i,t}$, then the signature is generated by the user i at time T . This methodology is essentially the same as Bichsel et al. [4] and its follow up works [12], [66]. Even though the opening opportunity is infrequent, this methodology requires $O(N)$ -times revocation check procedures for opening where N is the number of total users. Hence, we simply employ the ElGamal encryption that is employed to encrypt a user certificate (A in the scheme). Then, the opening cost is $O(1)$. For employing the ElGamal encryption scheme, we assume that the DDH problem is hard on \mathbb{G}_1 (which naturally holds since we employ Type-3 curves).

High Level Description of Our Revocation Methods: Before giving our scheme, we give a high level description of the natural revocation and premature revocation respectively. First, we give a high level description of the natural revocation as follows. The group manager has two signing keys of

the BBS+ signature scheme, γ_A and γ_B . Time information is managed by a binary tree BT with T leaf nodes where T is the maximum size of time. Let an expiry time τ be associated to a leaf node η . Let $\text{Path}(\eta) := (u_1, u_2, \dots, u_\ell)$, where u_1 is the root node, $u_\ell = \eta$, and $\ell = \log T$. Assume that each u_i is encoded in a \mathbb{Z}_p element. Then, a signer whose expiry time τ has a certificate $\{(A_j, \xi_j, \zeta_j)\}_{j \in [1, \ell]}$ as a signing key gsk

where $A_j = (gh_0^{\xi_j} h_1^{u_j} X)^{\frac{1}{\xi_j + \gamma_A}}$. Here, $X = h_2^x$, and $x = usk$ is known by the signer only (an output of the Join algorithm). ζ_j and ξ_j are random values, and g, h_0 , and h_1 are public values. Each A_j is a BBS+ signature of two messages, the node $u_j \in \text{Path}(\eta)$ and x . At the time t , the group manager runs the $\text{CS-TBK}(\text{BT}, t)$ algorithm, and let $Y := (v_1, v_2, \dots, v_{\text{num}})$ be the output of the algorithm. Expiration information ei_t contains $\{(B_{i,t}, \xi'_i, \zeta'_i)\}_{i \in [1, \text{num}]}$ where $B_{i,t} = (gh_0^{\xi'_i} h_1^{v_i} h_2^t)^{\xi'_i + \gamma_B}$.

Each $B_{i,t}$ is a BBS+ signature of two messages $v_i \in Y$ and the current time t . Due to the CS method, if $\tau < t$, then there exists a node $u \in \text{Path}(\eta) \cap Y$. So, a non-revoked signer can prove that there exist two signatures of the same node u contained in own gsk and ei_t respectively by using zero-knowledge proofs. Remark that if a signer whose expiry time $\tau > t$ tries to compute a valid group signature, then the signer needs to prepare the corresponding BBS+ signature B . This contradicts unforgeability of the BBS+ signature scheme, and thus the unforgeability of expiry time for signing keys is guaranteed.

Second, we give a high level description of the premature revocation as follows. In the Join/Issue phase, the group manager stores a revocation token $grt_i = \tilde{X}_i$ where $\tilde{X}_i = \tilde{g}^i \in \mathbb{G}_1$ and $x_i = usk_i$. At the time t , the group manager chooses $y_t \xrightarrow{\$} \mathbb{Z}_p$, and sets $\tilde{h}_t = \tilde{g}^{y_t}$ and $\hat{h}_t := \tilde{g}^{y_t} \in \mathbb{G}_2$. Then, $e(\tilde{h}_t, \tilde{g}) = e(\tilde{g}^{y_t}, \tilde{g}) = e(\tilde{g}, \tilde{g}^{y_t}) = e(\tilde{g}, \hat{h}_t)$ hold. A group signature σ contains $\tilde{h}_i^\beta, \tilde{g}^{d(x_i + \beta)}, \tilde{g}^d$ and \tilde{g}^d where β and d are randomness chosen by the signer. If a signer i is prematurely revoked, then the group manager computes $grt_{i,t} := grt_i^{y_t} = \tilde{h}_i^{x_i}$, and stores $grt_{i,t}$ to RL_t . Then, $grt_{i,t}$ satisfies $e(grt_{i,t}, \tilde{h}_i^\beta, \tilde{g}^d) = e(\tilde{h}_i^{x_i + \beta}, \tilde{g}^d) = e(\tilde{g}^{y_t(x_i + \beta)}, \tilde{g}^d) = e(\tilde{g}^{d(x_i + \beta)}, \tilde{g}^{y_t}) = e(\tilde{g}^{d(x_i + \beta)}, \hat{h}_t)$. By checking whether the equation holds for each $grt_{i,t}$ one by one, the verifier can check whether the signer is prematurely revoked or not. The randomness d , chosen in each signing, prevents to link two signatures generated by the same signer at the same time period.

As well as group signature schemes secure in the random oracle model, we employ signatures converted by the Fiat-Shamir transformation [67] which converts a three-move Σ protocol to NIZK proof by using a hash function (modeled as a random oracle in the security proof). Briefly, in the underlying Σ protocol, first a prover sends a commitment R , and a verifier returns the challenge c , and finally the prover sends a proof s . In the Fiat-Shamir transformation, the challenge c is computed by a hash function of the commitment and public values. If a message m is also hashed, then (s, c) can be seen as a signature on the message m . We call the signatures SPKs. We adopt the notation given by Nakanishi and Funabiki [28] as follows.

$$\text{SPK}\{(x_1, \dots, x_N) : R(x_1, \dots, x_N)\}(m).$$

Here x_1, \dots, x_N are N witnesses for some $N \in \mathbb{N}$, and R is a relation. That is, this is a signature on a message m generated by a signer who knows witnesses x_1, \dots, x_N which satisfy the

relation $R(x_1, \dots, x_N)$. In the random oracle model, it is well known that SPKs can be simulated without the knowledge (x_1, \dots, x_N) by using a simulator of the underlying proof of knowledge. Moreover, there exists an extractor that can extract the knowledge to be proved from two SPKs whose commitments are the same but the challenges are different.

Next, we give our scheme as follows.

Proposed GS-TBK scheme

- **GKeyGen**(λ): Choose $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g})$ where g and \hat{g} are generators of \mathbb{G}_1 and \mathbb{G}_2 respectively. Choose $f, \tilde{g}, g_2, h_0, h_1, h_2 \xleftarrow{\$} \mathbb{G}_1$, $\gamma_A, \gamma_B, \gamma_O \xleftarrow{\$} \mathbb{Z}_p$, and a hash function $H: \{0, 1\}^* \rightarrow \mathbb{Z}_p$ and $H': \{0, 1\}^* \rightarrow \mathbb{Z}_p$ (which are modeled as random oracles in the security proof). Set $vk_A = \hat{g}^{\gamma_A}$, $vk_B = \hat{g}^{\gamma_B}$, $g_1 = f^{\gamma_O}$, and $\text{reg} = \emptyset$. Output $gpk = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g}), f, \tilde{g}, g_1, g_2, h_0, h_1, h_2, vk_A, vk_B, H, H')$ and $msk = (\gamma_A, \gamma_B, \gamma_O)$.

- **Join**(gpk)/**Issue**(msk, reg, τ_i):

- A signer i chooses $x_i \xleftarrow{\$} \mathbb{Z}_p$, sets $usk_i = x_i$, computes $X_i = h_2^{x_i}$ and $\tilde{X}_i = \tilde{g}^{x_i}$, and sends (X_i, \tilde{X}_i) to the group manager. Moreover, the signer i proves the knowledge of x_i to the group manager as follows.

* The signer chooses $r_x \xleftarrow{\$} \mathbb{Z}_{\mathcal{O}}$, computes $R = h_2^{r_x}$, $\tilde{R} = \tilde{g}^{r_x}$, $c_x \leftarrow H'(X_i, \tilde{X}_i, R, \tilde{R})$, and $s_x = r_x + c_x x_i$, and sends (s_x, c_x) to the group manager.

* The group manager checks whether $c_x = H'(X_i, \tilde{X}_i, h_2^{s_x}/X_i^{c_x}, \tilde{g}^{s_x}/\tilde{X}_i^{c_x})$.

- The group manager assigns a leaf node η to τ_i . Remark that if the same time has been assigned to another signer before, then the same leaf node is selected. For all $u_j \in \text{Path}(\eta) := (u_1, u_2, \dots, u_\ell)$, the group manager computes BBS+ signatures $\{(A_j, \xi_j, \zeta_j)\}_{j \in [1, \ell]}$ where $A_j = (gh_0^{\xi_j} h_1^{u_j} X_i)^{\frac{1}{\xi_j + \gamma_A}}$, and sends $gsk_i = (\{(A_j, \xi_j, \zeta_j), u_i\}_{j \in [1, \ell]})$ and τ_i to the signer i .
- The group manager sets $grt_i = \tilde{X}_i$ and stores $(\tau_i, grt_i, \{(A_j, \xi_j, \zeta_j)\}_{j \in [1, \ell]})$ to $\text{reg}[i]$.
- **Revoke**($gpk, msk, t, \text{reg}, \text{RU}_t$): Choose $y_t \xleftarrow{\$} \mathbb{Z}_p$, and compute $\hat{h}_t = \tilde{g}^{y_t}$ and $\tilde{h}_t = \tilde{g}^{y_t}$.
 - **Generating Expiration Information**: For the current time t , obtain $Y := (v_1, v_2, \dots, v_{\text{num}}) \leftarrow \text{CS-TBK}(\text{BT}, t)$. Compute BBS+ signatures $\{(B_{i,t}, \xi'_i, \zeta'_i)\}_{i \in [1, \text{num}]}$ where $B_{i,t} = (gh_0^{\xi'_i} h_1^{v_i} h_2^t)^{\frac{1}{\xi'_i + \gamma_B}}$. Set $ei_t = (\hat{h}_t, \{(B_{i,t}, \xi'_i, \zeta'_i), v_i\}_{i \in [1, \text{num}]})$.
 - **Generating Revocation List**: For all $i \in \text{RU}_t$, compute $grt_{i,t} = grt_i^{y_t}$ and set $\text{RL}_t = (\hat{h}_t, \tilde{h}_t, \{grt_{i,t}\}_{i \in \text{RU}_t})$.

Output (ei_t, RL_t) .

- **Sign**($gpk, gsk_i, usk_i, m, t, ei_t$): Assume that $t < \tau_i$. Then, there exists a node u such that $((A, \xi, \zeta), u)$ is contained in gsk_i , where $A = (gh_0^{\xi} h_1^u X_i)^{\frac{1}{\xi + \gamma_A}}$, and $((B_t, \xi', \zeta'), u)$ is contained in ei_t , where $B_t = (gh_0^{\xi'} h_1^u h_2^t)^{\frac{1}{\xi' + \gamma_B}}$. Choose $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$\psi_1 = f^\alpha, \psi_2 = Ag_1^\alpha, \text{ and } \psi_3 = B_t g_2^\alpha.$$

Here, (ψ_1, ψ_2, ψ_3) is an ElGamal ciphertext with Kurosawa's randomness reuse technique [68].

Choose $\beta, d \xleftarrow{\$} \mathbb{Z}_p$ and compute

$$\psi_4 = \tilde{h}_t^\beta, \psi_5 = \tilde{g}^{d(x_i + \beta)}, \psi_6 = \tilde{g}^d, \text{ and } \psi_7 = \tilde{g}^d.$$

Here, $(\psi_4, \psi_5, \psi_6, \psi_7)$ is for the premature revocation check. Set $\delta = \alpha\xi$ and $\delta' = \alpha\xi'$, and compute a SPK V where

$$V = \text{SPK}\{(\alpha, \beta, \xi, \xi', \zeta', \zeta', u, x_i, \delta, \delta')\}$$

$$\frac{e(\psi_2, vk_A)}{e(g, \hat{g})} = \frac{e(h_0, \hat{g})^\xi e(h_1, \hat{g})^u e(h_2, \hat{g})^{x_i} e(g_1, vk_A)^\alpha e(g_1, \hat{g})^\delta}{e(\psi_2, \hat{g})^\xi}$$

$$\wedge \frac{e(\psi_3, vk_B)}{e(g, \hat{g})e(h_2, \hat{g})^t} = \frac{e(h_0, \hat{g})^{\xi'} e(h_1, \hat{g})^u e(g_2, vk_B)^\alpha e(g_2, \hat{g})^{\delta'}}{e(\psi_3, \hat{g})^{\xi'}}$$

$$\wedge \psi_1 = f^\alpha \wedge \psi_1^\xi f^{-\delta} = 1 \wedge \psi_1^{\xi'} f^{-\delta'} = 1 \wedge \psi_4 = \tilde{h}_t^\beta$$

$$\wedge \psi_5 = \psi_6^{x_i + \beta}(m),$$

as follows. Remark that the current time t is not hidden and is not a witness. Moreover, ψ_7 is not explicitly included in the statement of V since the validity of ψ_7 can be verified by checking $e(\psi_6, \hat{g}) = e(\tilde{g}, \psi_7)$ holds.

- Choose $r_\alpha, r_\beta, r_\xi, r_{\xi'}, r_{\zeta'}, r_{\zeta'}, r_u, r_x, r_\delta, r_{\delta'} \xleftarrow{\$} \mathbb{Z}_p$.
- Compute

$$R_1 = e(h_0, \hat{g})^{r_\xi} e(h_1, \hat{g})^{r_u} e(h_2, \hat{g})^{r_x} \times e(g_1, vk_A)^{r_\alpha} e(g_1, \hat{g})^{r_\delta} e(\psi_2, \hat{g})^{-r_\xi}$$

$$R_2 = e(h_0, \hat{g})^{r_{\xi'}} e(h_1, \hat{g})^{r_u} e(g_2, vk_B)^{r_\alpha} \times e(g_2, \hat{g})^{r_{\delta'}} e(\psi_3, \hat{g})^{-r_{\xi'}}$$

$$R_3 = \psi_1^{r_\xi} f^{-r_\delta}, R_4 = \psi_1^{r_{\xi'}} f^{-r_{\delta'}}$$

$$R_5 = \tilde{h}_t^{r_\beta}, R_6 = \psi_6^{r_x + r_\beta}.$$

- Compute $c \leftarrow H(\psi_1, \dots, \psi_7, R_1, \dots, R_6, m)$.
- Compute $s_\alpha = r_\alpha + c\alpha$, $s_\beta = r_\beta + c\beta$, $s_\xi = r_\xi + c\xi$, $s_{\xi'} = r_{\xi'} + c\xi'$, $s_{\zeta'} = r_{\zeta'} + c\zeta'$, $s_u = r_u + cu$, $s_x = r_x + cx_i$, $s_\delta = r_\delta + c\delta$, and $s_{\delta'} = r_{\delta'} + c\delta'$.

Output $\sigma = (\psi_1, \dots, \psi_7, c, s_\alpha, s_\beta, s_\xi, s_{\xi'}, s_{\zeta'}, s_{\zeta'}, s_u, s_x, s_\delta, s_{\delta'})$.

- **Verify**($gpk, t, \sigma, m, \text{RL}_t$): Parse $\text{RL}_t = (\hat{h}_t, \tilde{h}_t, \{grt_{i,t}\}_{i \in \text{RU}_t})$.
 - **Verification Check**: If $e(\psi_6, \hat{g}) \neq e(\tilde{g}, \psi_7)$, then output invalid. Otherwise, compute

$$R'_1 = e(h_0, \hat{g})^{s_\xi} e(h_1, \hat{g})^{s_u} e(h_2, \hat{g})^{s_x} e(g_1, vk_A)^{s_\alpha} \times e(g_1, \hat{g})^{s_\delta} e(\psi_2, \hat{g})^{-s_\xi} \left(\frac{e(\psi_2, vk_A)}{e(g, \hat{g})} \right)^{-c}$$

$$R'_2 = e(h_0, \hat{g})^{s_{\xi'}} e(h_1, \hat{g})^{s_u} e(g_2, vk_B)^{s_\alpha} e(g_2, \hat{g})^{s_{\delta'}} \times e(\psi_3, \hat{g})^{-s_{\xi'}} \left(\frac{e(\psi_3, vk_B)}{e(g, \hat{g})e(h_2, \hat{g})^t} \right)^{-c}$$

$$R'_3 = \psi_1^{s_\xi} f^{-s_\delta}, R'_4 = \psi_1^{s_{\xi'}} f^{-s_{\delta'}}$$

$$R'_5 = \tilde{h}_t^{s_\beta} \psi_4^{-c}, R'_6 = \psi_6^{s_x + s_\beta} \psi_5^{-c}.$$

If $c \neq H(\psi_1, \dots, \psi_7, R'_1, \dots, R'_6, m)$, then output invalid.

- **Revocation Check**: If there exists $grt_{i,t}$ such that $e(grt_{i,t} \psi_4, \psi_7) = e(\psi_5, \hat{h}_t)$ holds, then output invalid.

Otherwise, output valid.

- **Open**($gpk, msk, t, \text{reg}, \sigma, m, \text{RL}_t$): If invalid \leftarrow Verify($gpk, t, \sigma, m, \text{RL}_t$), then output \perp . Otherwise, parse

$\sigma = (\psi_1, \dots, \psi_7, c, s_\alpha, s_\beta, s_\zeta, s_\xi, s_{\zeta'}, s_{\xi'}, s_u, s_x, s_\delta, s_{\delta'})$ and $msk = (\gamma_A, \gamma_B, \gamma_O)$. Compute $A = \psi_2/\psi_1^{\gamma_O}$, search i such that $\text{reg}[i]$ contains A , and output i . If no such an entry exists, then output \perp .

Security Analysis. Here, we show that the proposed scheme is BU-anonymous, traceable, and non-frameable.

Theorem 4.1. *The proposed GS-TBK scheme satisfies BU-anonymity if the DDH1 assumption and the DLIN assumption hold in the random oracle model.*

We define the following games.

- *Game 0:* This is the same as the definition of BU-anonymity.
- *Game 1:* This game is the same as Game 0 except for the challenge signature is computed by programming of the random oracle H .
- *Game 2:* This game is the same as Game 0 except for the challenge signature $\sigma^* = (\psi_1^*, \psi_2^*, \psi_3^*, \psi_4^*, \psi_5^*, \psi_6^*, \psi_7^*, c^*, s_\alpha^*, s_\beta^*, s_\zeta^*, s_\xi^*, s_{\zeta'}^*, s_{\xi'}^*, s_u^*, s_x^*, s_\delta^*, s_{\delta'}^*)$, $\psi_2^*, \psi_3^* \xleftarrow{\$} \mathbb{G}_1$.
- *Game 3:* This game is the same as Game 1 except for the challenge signature $\sigma^* = (\psi_1^*, \psi_2^*, \psi_3^*, \psi_4^*, \psi_5^*, \psi_6^*, \psi_7^*, c^*, s_\alpha^*, s_\beta^*, s_\zeta^*, s_\xi^*, s_{\zeta'}^*, s_{\xi'}^*, s_u^*, s_x^*, s_\delta^*, s_{\delta'}^*)$, $\psi_5^* \xleftarrow{\$} \mathbb{G}_1$.

Let S_i be the event that \mathcal{A} successfully guesses b in Game i .

Lemma 4.1. $|\Pr[S_0] - \Pr[S_1]|$ is negligible.

Proof. In Game 1, for computing the challenge signature, first compute $\psi_1^*, \dots, \psi_7^*$ as in the scheme. Next, randomly choose $c, s_\alpha, s_\beta, s_\zeta, s_\xi, s_{\zeta'}, s_{\xi'}, s_u, s_x, s_\delta, s_{\delta'} \xleftarrow{\$} \mathbb{Z}_p$, and compute $R'_1 = e(h_0, \widehat{g})^{s_\zeta} e(h_1, \widehat{g})^{s_u} e(h_2, \widehat{g})^{s_x} e(g_1, vk_A)^{s_\alpha} e(g_1, \widehat{g})^{s_\beta} e(\psi_2, \widehat{g})^{-s_\zeta} \left(\frac{e(\psi_2, vk_A)}{e(g, \widehat{g})}\right)^{-c}$, $R'_2 = e(h_0, \widehat{g})^{s_{\zeta'}} e(h_1, \widehat{g})^{s_u} e(g_2, vk_B)^{s_\alpha} e(g_2, \widehat{g})^{s_{\beta'}} e(\psi_3, \widehat{g})^{-s_{\zeta'}} \left(\frac{e(\psi_3, vk_B)}{e(g, \widehat{g})e(h_2, \widehat{g})}\right)^{-c}$, $R'_3 = \psi_1^{s_\zeta} f^{-s_\beta}$, $R'_4 = \psi_1^{s_{\zeta'}}$, $f^{-s_{\beta'}}$, $R'_5 = \widehat{h}_t^{s_\beta} \psi_4^{-c}$, and $R'_6 = \psi_6^{s_\alpha + s_\beta} \psi_5^{-c}$. Next, programming the random oracle H such that $c := H(\psi_1, \dots, \psi_7, R'_1, \dots, R'_6, m)$, and send $\sigma = (\psi_1, \dots, \psi_7, c, s_\alpha, s_\beta, s_\zeta, s_\xi, s_{\zeta'}, s_{\xi'}, s_u, s_x, s_\delta, s_{\delta'})$ to \mathcal{A} . If programming fails (i.e., c collides with a value returned by H), output a random bit and aborts. If programming does not fail, $\Pr[S_0] = \Pr[S_1]$ holds. Since c is randomly chosen from \mathbb{Z}_p , the failure probability is at most q_h/p . \square

Lemma 4.2. $|\Pr[S_1] - \Pr[S_2]| \leq \text{Adv}_{\text{DDH1}}(\lambda)$.

Proof. Let $((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, f, \widehat{f}), f^a, f^b, Z)$ be a DDH1 instance. We construct an algorithm \mathcal{B} that distinguishes $Z = f^{ab}$ or not. \mathcal{B} implicitly sets $\alpha := a$ and $\gamma_O := b$ (thus, $g_1 = f^{\gamma_O} = f^b$). \mathcal{B} chooses $r \xleftarrow{\$} \mathbb{Z}_p$, and sets $g_2 := f^r$. \mathcal{B} chooses all values, except f, g_1 , and g_2 . Since \mathcal{B} has all secret values, \mathcal{B} can respond all queries issued by \mathcal{A} . In the challenge phase, \mathcal{B} selects (A, B_{t^*}) according to the scheme. \mathcal{B} sets $\psi_1^* := f^a$, $\psi_2^* := AZ$, and $\psi_3^* := B_{t^*} Z^r$. \mathcal{B} computes other components, except s_α^* is computed by programming of the random oracle H . If $Z = f^{ab}$, then \mathcal{B} correctly simulates Game 1, and if Z is a random value, then \mathcal{B} correctly simulates Game 2. \square

Lemma 4.3. $|\Pr[S_2] - \Pr[S_3]| \leq \text{Adv}_{\text{DLIN}}(\lambda)(1/q_A q_R - q_s q_h/p)$ where q_A, q_R, q_s , and q_h are the number of AddU, Revoke, GSign, and hash queries respectively.

Proof. Let $((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \widehat{g}, \widehat{g}), g', \widehat{g}', h, \widehat{h}, \widehat{g}^a, g^b, Z)$ be a DLIN instance. We construct an algorithm \mathcal{B} that distinguishes $Z = h^{a+b}$ or not. \mathcal{B} guesses when the challenge user, say i^* , is added in the group with the probability $1/q_A$, and guesses the challenge time t^* with the probability $1/q_R$. We assume that the guesses are correct. \mathcal{B} chooses $y_{t^*} \xleftarrow{\$} \mathbb{Z}_p$, and implicitly sets $x_{i^*} := a$, $\beta^* := b$, and $y_{t^*} := y_{t^*}^c$ where $g' := \widehat{g}^c$ and $\widehat{g}' := \widehat{g}^c$ for some $c \in \mathbb{Z}_p$. \mathcal{B} chooses $y_t \xleftarrow{\$} \mathbb{Z}_p$ for $t \neq t^*$. Then, \mathcal{B} can compute $(\widehat{h}_t, \widehat{h}_t)$ as follows.

$$(\widehat{h}_t, \widehat{h}_t) = \begin{cases} (\widehat{g}^{y_t}, \widehat{g}^{y_t}) & (t \neq t^*) \\ ((g')^{y_{t^*}}, (\widehat{g}')^{y_{t^*}}) & (t = t^*). \end{cases}$$

Since \mathcal{B} has all secret keys of signers, except i^* 's one, \mathcal{B} can respond all queries issued by \mathcal{A} if these are not related to i^* . Hence, we show the simulation of revocation queries $\text{Revoke}(\text{RU}_t)$ where $i^* \in \text{RU}_t$, and signing queries $\text{GSign}(i^*, m)$.

For revocation queries, \mathcal{B} can revoke i^* by computing $\text{grt}_{i^*, t} = (\widehat{g}^a)^{y_t}$ when $t \neq t^*$. Remark that \mathcal{B} is not required to compute grt_{i^*, t^*} since i^* is not revoked at the challenge time t^* . This leads to backward unlinkability.

For signing queries at $t = t^*$, \mathcal{B} randomly chooses $\psi_1, \dots, \psi_3 \xleftarrow{\$} \mathbb{G}_1$ and $\beta, d \xleftarrow{\$} \mathbb{Z}_p$, and computes $\psi_4 = \widehat{h}_{t^*}^\beta$. Then $\psi_4^{1/y_{t^*}} = ((\widehat{g}^{y_{t^*}})^{\beta})^{1/y_{t^*}} = \widehat{g}^\beta$ hold. \mathcal{B} computes $\psi_5 = (g^a \widehat{g}^\beta)^d$, $\psi_6 = \widehat{g}^d$, and $\psi_7 = \widehat{g}^d$. Now, the revocation check relation $e(\psi_5, \widehat{h}_{t^*}) = e((g^a \widehat{g}^\beta)^d, \widehat{g}^{y_{t^*}}) = e((\widehat{g}^a \psi_4^{1/y_{t^*}})^d, \widehat{g}^{y_{t^*}}) = e((\widehat{g}^{y_{t^*}})^{x_{i^*}} \psi_4, \widehat{g}^d) = e(\text{grt}_{i^*, t^*} \psi_4, \psi_7)$ holds. Other components are computed by programming of the random oracle H .

For signing queries at $t \neq t^*$, \mathcal{B} randomly chooses $\psi_1, \dots, \psi_4 \xleftarrow{\$} \mathbb{G}_1$ and $d \xleftarrow{\$} \mathbb{Z}_p$, and computes $\psi_5 := (\widehat{g}^a \cdot \psi_4^{1/y_t})^d$, $\psi_6 = \widehat{g}^d$, and $\psi_7 = \widehat{g}^d$. Then, the revocation check relation $e(\psi_5, \widehat{h}_t) = e((\widehat{g}^a \cdot \psi_4^{1/y_t})^d, \widehat{g}^{y_t}) = e((\widehat{g}^a)^{y_t} \psi_4, \widehat{g}^d) = e(\text{grt}_{i^*, t} \psi_4, \psi_7)$ holds. Other components are computed by programming of the random oracle H .

For computing the challenge signature, \mathcal{B} chooses $\psi_1^*, \psi_2^*, \psi_3^* \xleftarrow{\$} \mathbb{G}_1$, computes $\psi_4^* = (g^b)^{y_{t^*}} = (\widehat{g}^b)^{y_{t^*}} = (\widehat{g}^{y_{t^*}})^b = \widehat{h}_{t^*}^{\beta^*}$, and sets $\psi_5^* = Z$, $\psi_6^* = h$, and $\psi_7^* = \widehat{h}$. Other components are computed by programming of the random oracle H . If $Z = h^{a+b}$, then \mathcal{B} correctly simulates Game 2, and if Z is a random value, then \mathcal{B} correctly simulates Game 3. \square

Since now the challenge signature does not depend on the challenge bit, $\Pr[S_3] = 1/2$. This concludes the proof.

Theorem 4.2. *The proposed GS-TBK scheme satisfies traceability in the random oracle model under the q -SDH assumption and the knowledge of secret key assumption.*

As in the Ohara et al. scheme, we introduce the KOSK assumption [33] where the adversary is required to reveal the secret key of the honest users. The reason why we need to introduce the assumption is explained as follows. In the Join algorithm, a user sends $X_i = h_2^{x_i}$ (and $\widehat{X}_i = \widehat{g}^{x_i}$ also). The group manager signs x_i by using the signing key of the BBS+ signature scheme such that $A_j = (gh_0^{\xi_j} h_1^{u_j} X_i)^{\frac{1}{\xi_j + \gamma_A}}$. Due

to the form of the BBS+ signature scheme, the group manager can sign x_i without knowing x_i . On the other hand, in the security proof, the simulator needs to send a signed message x_i in order to send a signing query to the signing oracle of the underlying BBS+ signature scheme. So, we use the KOSK assumption.

We can construct an algorithm that extracts a BBS+ signature by applying the Forking lemma [69], [70]. More precisely, from the winning condition $i \notin \text{CU} \setminus \text{RU}_{i^*}$, an adversary needs to produce a forged group certificate A that is not issued via the SndToU oracle, or needs to produce a forged certificate of non-revoked signers B that is not generated when the Revoke oracle is called. Since the signature output by the adversary is valid, forged BBS+ signatures are extracted from the signature. The unforgeability of expiry time for signing keys is also reduced to unforgeability of the BBS+ signature scheme. That is, if an adversary can produce a valid signature though an expiry time τ_i has passed, i.e., $\tau_i < t^*$, then there exists a BBS+ signature B which is valid and is not contained in RL_{t^*} . So, we can construct an algorithm that extracts such B by applying the Forking lemma. Thus, if the extraction works well, then Theorem 4.2 holds. We prove that the following lemma for these extractions.

Lemma 4.4. *The SPK V proves the knowledge $\alpha, \beta, \zeta, \xi, \zeta', \xi', u,$*

$$x_i, \delta, \delta' \text{ such that } \psi_1 = f^\alpha, \psi_2 = (gh_0^\zeta h_1^u h_2^{x_i} g_1^{\gamma_A \alpha + \delta})^{\frac{1}{\xi + \gamma_A}}, \psi_3 = (gh_0^{\zeta'} h_1^u h_2^{x_i} g_2^{\gamma_B \alpha + \delta'})^{\frac{1}{\xi' + \gamma_B}}, \psi_4 = \tilde{h}_t^\beta, \text{ and } \psi_5 = \psi_6^{x_i + \beta}.$$

Proof. By the knowledge extractor for V , we can obtain $\alpha, \beta, \zeta, \xi, \zeta', \xi', u, x_i, \delta, \delta'$ such that

$$\frac{e(\psi_2, vk_A)}{e(g, \hat{g})} = \frac{e(h_0, \hat{g})^\zeta e(h_1, \hat{g})^u e(h_2, \hat{g})^{x_i} e(g_1, vk_A)^\alpha e(g_1, \hat{g})^\delta}{e(\psi_2, \hat{g})^\xi} \quad (1)$$

$$\frac{e(\psi_3, vk_B)}{e(g, \hat{g})e(h_2, \hat{g})^t} = \frac{e(h_0, \hat{g})^{\zeta'} e(h_1, \hat{g})^u e(g_2, vk_B)^\alpha e(g_2, \hat{g})^{\delta'}}{e(\psi_3, \hat{g})^{\xi'}} \quad (2)$$

$$\psi_1 = f^\alpha \quad (3)$$

$$\psi_1^\xi f^{-\delta} = 1 \quad (4)$$

$$\psi_1^{\xi'} f^{-\delta'} = 1 \quad (5)$$

$$\psi_4 = \tilde{h}_t^\beta \quad (6)$$

$$\psi_5 = \psi_6^{x_i + \beta}. \quad (7)$$

From (1), the equation

$$e(\psi_2, vk_A \hat{g}^\delta) = e(h_0^\zeta h_1^u h_2^{x_i}, \hat{g}) e(g_1, vk_A \hat{g}^\delta) e(g, \hat{g}),$$

holds. Set $\psi_2 = g^\theta$, $h_0 = g_1^{\theta_0}$, $h_1 = g_1^{\theta_1}$, $h_2 = g_1^{\theta_2}$, and $g_1 = g^\mu$ for some $\theta, \theta_1, \theta_2, \mu \in \mathbb{Z}_p$. Since $vk_A = \hat{g}^{\gamma_A}$, $e(g, \hat{g})^{\theta(\xi + \gamma_A)} = e(g, \hat{g})^{\mu(\theta_0 \zeta + \theta_1 u + \theta_2 x_i + \gamma_A \alpha + \delta) + 1}$, and thus $\theta(\xi + \gamma_A) = \mu(\theta_0 \zeta + \theta_1 u + \theta_2 x_i + \gamma_A \alpha + \delta) + 1 \pmod p$ holds. This means $\psi_2 = g^\theta = (g^{\mu(\theta_0 \zeta + \theta_1 u + \theta_2 x_i + \gamma_A \alpha + \delta) + 1})^{\frac{1}{\xi + \gamma_A}} = (gh_0^\zeta h_1^u h_2^{x_i} g_1^{\gamma_A \alpha + \delta})^{\frac{1}{\xi + \gamma_A}}$ holds. Similarly, from (2), $\psi_3 = (gh_0^{\zeta'} h_1^u h_2^{x_i} g_2^{\gamma_B \alpha + \delta'})^{\frac{1}{\xi' + \gamma_B}}$ holds. From (3), the extracted α satisfies $\psi_1 = f^\alpha$. Then,

from (4) and (5), $\delta = \alpha \xi$ and $\delta' = \alpha \xi'$ holds. Finally, from (6) and (7), the extracted x_i and β satisfy $\psi_4 = \tilde{h}_t^\beta$ and $\psi_5 = \psi_6^{x_i + \beta}$. \square

Theorem 4.3. *The proposed GS-TBK scheme satisfies non-frameability in the random oracle model under the DL assumption.*

Proof. Let $(\mathbb{G}_1, \mathbb{G}_2, \tilde{\mathbb{G}}_T, e, \tilde{g}, \tilde{g}^x)$ be a DL instance. We construct an algorithm \mathcal{B} that breaks the DL problem as follows. Let q_A be the number of SndToU queries. \mathcal{B} guesses the user $i^* \in [1, q_A]$ that \mathcal{A} outputs in the final phase. We assume the guess is correct with the probability $1/q_A$. \mathcal{B} chooses $\theta_2 \xleftarrow{\$} \mathbb{Z}_p$ and sets $h_2 = \tilde{g}^{\theta_2}$, $usk_{i^*} := x$, and $X_{i^*} := (\tilde{g}^x)^{\theta_2} = h_2^x$, and $\tilde{X}_{i^*} := \tilde{g}^x$. \mathcal{B} chooses other all values as in the scheme. When i^* is added to the group via the SndToU query, \mathcal{B} chooses $s_x, c_x \xleftarrow{\$} \mathbb{Z}_p$, sets $c_x := H'(X_{i^*}, \tilde{X}_{i^*}, h_2^{s_x} / X_{i^*}^{c_x}, \tilde{g}^{s_x} / \tilde{X}_{i^*}^{c_x})$, and sends (s_x, c_x) to \mathcal{A} . For a signing query (\cdot, i^*) , \mathcal{B} programs the random oracle H and computes a signature. Finally, \mathcal{A} outputs a signature. \mathcal{B} rewinds \mathcal{A} and extracts x^* from the signatures output by \mathcal{A} by applying the Forking lemma [69], [70]. Since the signatures are traced to i^* , the extracted x^* satisfies $X_{i^*} = h_2^{x^*}$. So, \mathcal{B} outputs x^* if the extraction works well. \square

5 IMPLEMENTATION

In this section, we provide our implementation results. We set the maximum size of time T at 2,048, and each day is assigned to a leaf node (thus $\log_2 T = 11$). This setting is the same as that of Liu et al. [31]. Our implementation environment is as follows: CPU: Xeon E5-2660 v3 @ 2.60GHz, and gcc 4.9.2. We use the RELIC library (ver. 0.4.1)⁴ [41] for elliptic curve operations and the pairing operation, and OpenSSL (ver. 1.0.2d) [71] for cryptographically hashing (we employ SHA-512).

5.1 Reconsider the Order size of Elliptic Curves

As mentioned in Section 1.5, we selected a BN curve over a 254-bit prime field in the proceedings version [1], which was believed to have 128-bit security. However, recent progress in discrete logarithm problems of finite fields asymptotically reduces the complexity [52], [53], and recently, Menezes, Sarkar, and Singh evaluated that the 254-bit BN curve has at least 108-bit security. From their evaluation, it is necessary to use BN curves on the order of around 382-bit to ensure 128-bit security. Most recently, Barbulescu and Duquesne also reported their security evaluation, which is based on some heuristic assumptions from the top-record discrete logarithm computations, and showed that the 254-bit BN curve ensures no more than 100-bit security and the curve over a 462-bit prime field has 132-bit security [40]. In addition, they also showed that a BLS curve of embedding degree 12 over a 461-bit prime field has 131-bit security. In this full version, we follow the Barbulescu-Duquesne's evaluation, and select a 455-bit BLS curve of embedding degree 12 to ensure almost 128-bit security, which is supported by the RELIC library. In this setting, the sizes of the scalar value in \mathbb{Z}_p , an element in

4. Building with options `-DARITH=x64-asm-455 -DFP_PRIME=455 -DFPX_METHD=INTEG;INTEG;LAZYR -DPP_METHD=LAZYR;OATEP.`

TABLE 2
Benchmarks of Group Operations on
a 455-Bit BLS Curve

Operation	Time (μsec)	
Construct Precomp Table in \mathbb{G}_1	260.9	
Construct Precomp Table in \mathbb{G}_2	963.9	
Mul(\mathbb{G}_1, U)	373.3	
Mul(\mathbb{G}_1, K)	201.2	
Mul(\mathbb{G}_2, U)	821.2	
Mul(\mathbb{G}_2, K)	502.6	
Exp(\mathbb{G}_T, U)	1180.9	
Pairing	Miller loop	1046.0
	Final exp.	1193.5
	Total	2239.6

\mathbb{G}_1 , an element in \mathbb{G}_2 , and an element in \mathbb{G}_T are 39 bytes, 58 bytes, 115 bytes, and 456 bytes, respectively. Then the signature size is 892 bytes, and the size of expiration information is $58 + 144|\text{CS-TBK}(\text{BT}, t)|$ bytes (the maximum size is 1,642 bytes when $|\text{CS-TBK}(\text{BT}, t)| = \log_2 T = 11$). Table 2 summarizes benchmarks of elliptic curve and pairing operations in our environment.

Here, $\text{Mul}(\mathbb{G}_1, \text{Type})$, $\text{Mul}(\mathbb{G}_2, \text{Type})$, and $\text{Exp}(\mathbb{G}_T, \text{Type})$ are scalar multiplication on \mathbb{G}_1 and \mathbb{G}_2 , and exponentiation on \mathbb{G}_T , respectively. If a base point is previously known and fixed, then Type is set as K, and U otherwise. We note that we always use Type U for exponentiations on \mathbb{G}_T since the RELIC library does not support Type K on \mathbb{G}_T .

5.2 Reconsider the Computation of Our Algorithms

As mentioned in Section 1.5, we reconsider the computations of our algorithms according to the benchmarks of RELIC library (Table 2). Here we briefly describe our modifications of the computations. We use bilinearity of pairing to reduce the number of exponentiations on \mathbb{G}_T . For Sign, we modify R_1 and R_2 as

$$R_1 = e(h_0^{r_\xi} h_1^{r_u} h_2^{r_x} g_1^{r_\delta} \psi_2^{-r_\xi}, \hat{g}) e(g_1^{r_\alpha}, vk_A),$$

$$R_2 = e(h_0^{r_\xi'} h_1^{r_u} g_2^{r_\delta'} \psi_3^{-r_\xi'}, \hat{g}) e(g_2^{r_\alpha}, vk_B).$$

Here, we use multiplicative notations due to the readability. Original computations require 10 $\text{Exp}(\mathbb{G}_T, U)$ + 2 pairings (with precomputed pairing values), but our modifications require 8 $\text{Mul}(\mathbb{G}_1, K)$ + 2 $\text{Mul}(\mathbb{G}_1, U)$ + 4 pairings. The 4 pairings in our modifications can be computed by 4 Miller loop

TABLE 4
Benchmark of Each Algorithms of Our GS-TBK Scheme

Algorithm	Time (msec)
GKeyGen	2.830
Join	0.814
Issue	9.284
Sign	11.331 (Opt.)/20.281 (Original)
Revoke	8.823 (e_i) [†] /0.369 (RL) [‡]

[†]:The worst case when we set $T = 2,048$.

[‡]:For prematurely revoking one signer.

+ 2 Final exp. by sharing final exponentiations on the product of pairing values. Due to our optimization, we can reduce the running time of the Sign algorithm by approximately 45 percent compared to the original Sign algorithm. See Table 4. For Verify, we modify R'_1 and R'_2 as

$$R'_1 = e(h_0^{s_\xi} h_1^{s_u} h_2^{s_x} g_1^{s_\delta} \psi_2^{-s_\xi} g_1^c, \hat{g}) e(g_1^{s_\alpha} \psi_2^{-c}, vk_A),$$

$$R'_2 = e(h_0^{s_\xi'} h_1^{s_u} g_2^{s_\delta'} \psi_3^{-s_\xi'} g_1^c h_2^c, \hat{g}) e(g_2^{s_\alpha} \psi_3^{-c}, vk_B),$$

which originally requires 13 $\text{Exp}(\mathbb{G}_T, U)$ + 4 pairings (with precomputed pairing values), but 9 $\text{Mul}(\mathbb{G}_1, K)$ + 4 $\text{Mul}(\mathbb{G}_1, U)$ + 4 pairings are required in our modifications. Again, the 4 pairings can be computed by 4 Miller loop + 2 Final exp. Due to our optimization, we can reduce the running time of the verification check by approximately 50 percent compared to the original verification check. We summarize the number of operations for each algorithms in Table 3.

5.3 Benchmarks

Next we show benchmarks of algorithms, except Verify, in Table 4. We re-implement the proposed scheme by employing a 455-bit BLS curve of embedding degree 12. It is particularly worth noting that the computational cost of our Sign algorithm is constant in terms of both the time representation and the number of revoked signers. Our optimization, given in Section 5.2, allows us to reduce approximately 45 percent of the running time of the Sign algorithm.

Next, we show the benchmark of the Verify algorithm. Remark that in the usual VLR group signature schemes, the cost of the verification algorithm (more precisely the revocation check) linearly depends on the number of total revoked signers whereas in GS-TBK it just linearly depends on the number of prematurely revoked signers due to time-bound keys. Thus, we show the running time of the Verify algorithm for several numbers of prematurely revoked signers

TABLE 3
The Number of Operations for Each Algorithms of Our GS-TBK Scheme

Algorithm	Operations
GKeyGen	1 $\text{Mul}(\mathbb{G}_1, K)$ + 2 $\text{Mul}(\mathbb{G}_2, K)$ (+ 6 random point pickings)
Join	4 $\text{Mul}(\mathbb{G}_1, K)$ + 1 Hash
Issue	$(\log_2 T + 3)$ $\text{Mul}(\mathbb{G}_1, U)$ + $(2 \log_2 T + 4)$ $\text{Mul}(\mathbb{G}_1, K)$ + 1 Hash
Revoke-Expiration info.	$(4L + 1)$ $\text{Mul}(\mathbb{G}_1, K)$
Revoke-Revocation list	R_{pre} $\text{Mul}(\mathbb{G}_1, U)$
Sign	4 $\text{Mul}(\mathbb{G}_1, U)$ + 16 $\text{Mul}(\mathbb{G}_1, K)$ + 1 $\text{Mul}(\mathbb{G}_2, K)$ + 4 Miller loop + 2 Final exp. + 1 Hash
Verify-Verification check	10 $\text{Mul}(\mathbb{G}_1, U)$ + 11 $\text{Mul}(\mathbb{G}_1, K)$ + 6 Miller loop + 3 Final exp. + 1 Hash
Verify-Revocation check	$(R_{\text{pre}} + 1)$ Miller loop + $(R_{\text{pre}} + 1)$ Final exp.

L : the number of nodes outputted by CS-TBK(BT, t), at most $\log_2 T$.

R_{pre} : the number of prematurely revoked signers.

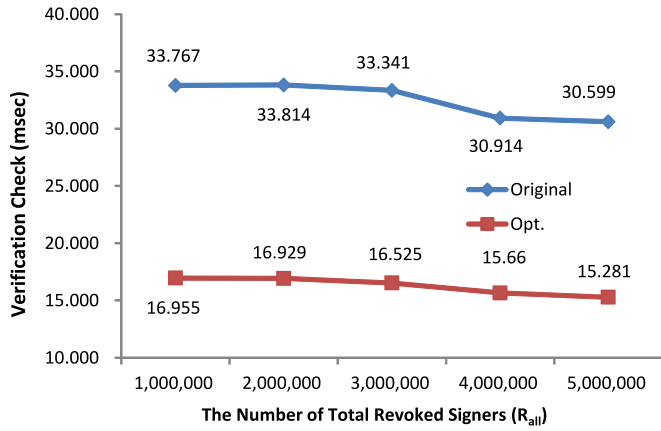


Fig. 3. The running time of the verification check.

R_{pre} . Recall that the Verify algorithm consists of two sub procedures, the verification check and the revocation check. The former is independent of R_{pre} whereas the latter depends on R_{pre} . Let

$$R_{all} := R_{pre} + R_{natural},$$

be the total number of revoked signers, where $R_{natural}$ be the number of naturally revoked signers, and we set

$$\text{Rate} := R_{pre}/R_{all}.$$

For example, when $R_{all} = 1,000,000$ and $\text{Rate} = 0.2$, then $R_{pre} = 200,000$ and $R_{natural} = 800,000$.

First, we show the running time of the verification check, i.e., the running time of the Verify algorithm with $\text{Rate} = 0$ in Fig. 3. The running time is approximately 16 msec regardless of R_{all} . Moreover, our optimization, given in Section 5.2, allows us to reduce approximately 50 percent of the running time of the verification check.

Next, we show the Verify algorithm for each Rate. We set $R_{all} = 5,000,000$ and show the running time of the Verify algorithm in Fig. 4. Since the revocation check requires $O(R_{pre})$ -times pairing computations⁵, the running time of the Verify algorithm linearly depends on R_{pre} . Nevertheless, as a reasonable assumption, the natural revocation accounts for most of signer revocations in practice and prematurely revoked signers are only a small fraction. Thus, for a relatively small Rate, our scheme is still efficient in practice.

As a reference, we show the running time of the Verify algorithm for each Rate and R_{all} in Fig. 5.

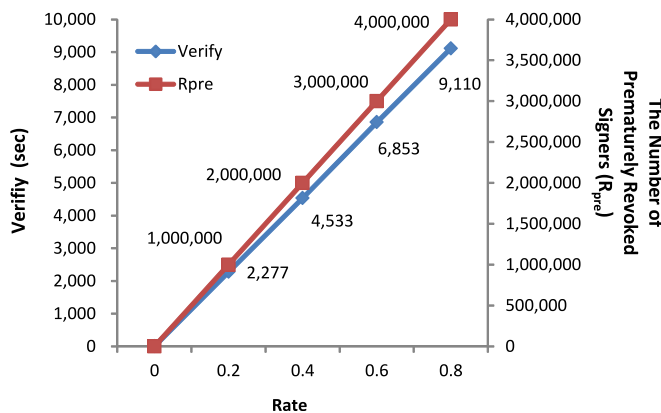


Fig. 4. The running time of the Verify algorithm ($R_{all} = 5,000,000$).



Fig. 5. The running time of the Verify algorithm (Rate = 0.2, 0.4, 0.6, 0.8).

6 CONCLUSION

In this paper, we revisit the definition of GS-TBK given in [30], [31], and give a new security model that considers the unforgeability of expiry time for signing keys. Moreover, the computational cost of our signing algorithm is constant whereas those of the previous schemes depend on the bit-length of the time representation.

In the proceedings version [1], we chose a BN curve over a 254-bit prime field which was believed to have 128-bit security. In this full version, due to the recent progress in discrete logarithm problems of finite fields [52], [53] and of security evaluations [39], [40], we select a BLS curve of embedding degree 12 over a 455-bit prime field to ensure almost 128-bit security. Moreover, we reconsider the computation of our algorithms and give implementations with these modifications. Our optimization allows us to reduce the running time of the signing algorithm by approximately 45 percent, and also reduce the running time of the verification check by approximately 50 percent.

We have mainly three drawbacks to be considered further. The first one is that signers need to download expiration information ei_t at each time t . The second one is the KOSK assumption. The third one is the security loss caused by the use of the forking lemma [69], [70]. For example, as mentioned by Ohara et al. [21], the advantage $\text{Adv}_{\text{GS-TBK},A}^{\text{nf}}(\lambda)$ is bounded by roughly $(\text{Adv}_{\text{DL}}(\lambda))^{1/2}$ (with some coefficients determined by the number of queries) due to the Bellare-Neven general forking lemma [69]. Then, due to the exact security [72], [73], if 128-bit security is required, then the probability of solving the DL problem needs to be less than 2^{-256} , and currently it is not clear and still under discussion how large order should be selected for such a strong security level. In order to avoid such a security loss due to a loose reduction, a tight reduction is highly desirable. We leave these as future works of this paper.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers of IEEE-TDSC for their invaluable comments and suggestions. This work was partially supported by JSPS KAKENHI Grant Number JP16K00198. An extended abstract appeared

⁵ In the previous schemes [30], [31], no pairing computation is required for the revocation check (just $O(R_{pre})$ -times exponentiations are required). Thus, our revocation check is inefficient than those of the previous schemes due to the pairing computations. However, at the expense of this inefficiency, our scheme provides backward unlinkability.

at the 12th ACM Asia Conference on Computer and Communications Security (ASIACCS 2017) [1]. In this journal version, we mainly reconsider two parts: the security parameters and the pairing equations in our algorithms, and re-implement the proposed scheme. See Section 1.5 Differences from the Proceedings Version.

REFERENCES

- [1] K. Emura, T. Hayashi, and A. Ishida, "Group signatures with time-bound keys revisited: A new model and an efficient construction," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, 2017, pp. 777–788.
- [2] D. Chaum and E. van Heyst, "Group signatures," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 1991, pp. 257–265.
- [3] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Annu. Int. Cryptology Conf.*, 2004, pp. 41–55.
- [4] P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi, "Get shorty via group signatures without encryption," in *Proc. 7th Int. Conf. Security Cryptography Netw.*, 2010, pp. 381–398.
- [5] X. Boyen and B. Waters, "Full-domain subgroup hiding and constant-size group signatures," in *Proc. Int. Workshop Public Key Cryptography*, 2007, pp. 1–15.
- [6] B. Libert, T. Peters, and M. Yung, "Short group signatures via structure-preserving signatures: Standard model security from simple assumptions," in *Proc. Annu. Int. Cryptology Conf.*, 2015, pp. 296–316.
- [7] B. Libert and D. Vergnaud, "Group signatures with verifier-local revocation and backward unlinkability in the standard model," in *Proc. 7th Int. Conf. Cryptology Netw. Security*, 2009, pp. 498–517.
- [8] B. Libert, F. Mouhartem, T. Peters, and M. Yung, "Practical "Signatures with efficient protocols" from simple assumptions," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, 2016, pp. 511–522.
- [9] J. Groth, "Fully anonymous group signatures without random oracles," in *Proc. Adv. Cryptology 13th Int. Conf. Theory Appl. Cryptology Inf. Security*, 2007, pp. 164–180.
- [10] J. Furukawa and H. Imai, "An efficient group signature scheme from bilinear maps," *Inst. Electron. Inf. Commun. Eng. Trans.*, vol. 89-A, no. 5, pp. 1328–1338, 2006.
- [11] C. Delerablée and D. Pointcheval, "Dynamic fully anonymous short group signatures," in *Proc. 1st Int. Conf. Cryptology*, 2006, pp. 193–210.
- [12] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Proc. Cryptographers Track. RSA Conf.*, 2016, pp. 111–126.
- [13] S. D. Gordon, J. Katz, and V. Vaikuntanathan, "A group signature scheme from lattice assumptions," in *Proc. Adv. Cryptology 13th Int. Conf. Theory Appl. Cryptology Inf. Security*, 2010, pp. 395–412. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17373-8_23
- [14] B. Libert, S. Ling, K. Nguyen, and H. Wang, "Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2016, pp. 1–31.
- [15] N. Attrapadung, K. Emura, G. Hanaoka, and Y. Sakai, "A revocable group signature scheme from identity-based revocation techniques: Achieving constant-size revocation list," in *Proc. Int. Conf. Appl. Cryptography Netw. Security*, 2014, pp. 419–437.
- [16] N. Attrapadung, K. Emura, G. Hanaoka, and Y. Sakai, "Revocable group signature with constant-size revocation list," *Comput. J.*, vol. 58, no. 10, pp. 2698–2715, 2015.
- [17] B. Libert, T. Peters, and M. Yung, "Group signatures with almost-for-free revocation," in *Proc. Annu. Int. Cryptology Conf.*, 2012, pp. 571–589.
- [18] B. Libert, T. Peters, and M. Yung, "Scalable group signatures with revocation," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2012, pp. 609–627.
- [19] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki, "Revocable group signature schemes with constant costs for signing and verifying," in *Proc. Int. Workshop Public Key Cryptography*, 2009, pp. 463–480.
- [20] T. Nakanishi and N. Funabiki, "Revocable group signatures with compact revocation list using accumulators," *Inst. Electron. Inf. Commun. Eng. Trans.*, vol. 98-A, no. 1, pp. 117–131, 2015.
- [21] K. Ohara, K. Emura, G. Hanaoka, A. Ishida, K. Ohta, and Y. Sakai, "Shortening the libert-peters-yung revocable group signature scheme by using the random oracle methodology," *Int. Association Cryptologic Res. Cryptology ePrint Archive*, vol. 2016, 2016, Art. no. 477.
- [22] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Proc. Cryptographers Track. RSA Conf.*, 2005, pp. 275–292.
- [23] S. Zhou and D. Lin, "Shorter verifier-local revocation group signatures from bilinear maps," in *Proc. Int. Conf. Cryptology Netw. Security*, 2006, pp. 126–143.
- [24] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *CCSP. 23rd ACM SIGSAC Conf. Comput. Commun. Security*, 2004, pp. 168–177.
- [25] J. Bringer and A. Patey, "VLR group signatures-how to achieve both backward unlinkability and efficient revocation checks," in *Proc. Int. Conf. Security Cryptography*, 2012, pp. 215–220.
- [26] S. Canard, G. Fuchsbauer, A. Gouget, and F. Laguillaumie, "Plaintext-checkable encryption," in *Proc. Cryptographers Track. RSA Conf.*, 2012, pp. 332–348.
- [27] T. Nakanishi and N. Funabiki, "Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps," in *Proc. Adv. Cryptology 13th Int. Conf. Theory Appl. Cryptology Inf. Security*, 2005, pp. 533–548.
- [28] T. Nakanishi and N. Funabiki, "A short verifier-local revocation group signature scheme with backward unlinkability," in *Proc. Int. Workshop Security*, 2006, pp. 17–32.
- [29] L. Wei and J. Liu, "Shorter verifier-local revocation group signature with backward unlinkability," in *Pairing-Based Cryptography*. Berlin, Heidelberg: Springer, 2010, pp. 136–146.
- [30] C. Chu, J. K. Liu, X. Huang, and J. Zhou, "Verifier-local revocation group signatures with time-bound keys," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, 2012, pp. 26–27.
- [31] J. K. Liu, C. Chu, S. S. M. Chow, X. Huang, M. H. Au, and J. Zhou, "Time-bound anonymous authentication for roaming networks," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 178–189, Jan. 2015.
- [32] H. Lin and W. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," in *Proc. Int. Conf. Appl. Cryptography Netw. Security*, 2005, pp. 456–466.
- [33] T. Ristenpart and S. Yilek, "The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2007, pp. 228–245.
- [34] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," *CRYPTO*, pp. 41–62, 2001.
- [35] Y. Dodis and N. Fazio, "Public key broadcast encryption for stateless receivers," in *DRMProc. ACM Workshop Digital Rights Manage.*, 2002, pp. 61–80.
- [36] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," in *Proc. Int. Workshop Public Key Cryptography*, 2003, pp. 31–46.
- [37] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2006, pp. 465–485.
- [38] P. S. L. M. Barreto, B. Lynn, and M. Scott, "Constructing elliptic curves with prescribed embedding degrees," in *Proc. Int. Conf. Security Commun. Netw.*, 2002, pp. 257–267.
- [39] A. Menezes, P. Sarkar, and S. Singh, "Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography," *Int. Association Cryptologic Res. Cryptology ePrint Archive*, vol. 2016, 2016, Art. no. 1102.
- [40] R. Barbulescu and S. Duquesne, "Updating key size estimations for pairings," *Int. Association Cryptologic Res. Cryptology ePrint Archive*, vol. 2017, 2017, Art. no. 334.
- [41] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an efficient library for cryptography." [Online]. Available: <https://github.com/relic-toolkit/relic>
- [42] V. Benjumea, J. Lopez, J. A. Montenegro, and J. M. Troya, "A first approach to provide anonymity in attribute certificates," in *Proc. Int. Workshop Public Key Cryptography*, 2004, pp. 402–415.
- [43] V. Benjumea, S. G. Choi, J. Lopez, and M. Yung, "Anonymity 2.0 - X.509 extensions supporting privacy-friendly authentication," in *Proc. Int. Conf. Cryptology Netw. Security*, 2007, pp. 265–281.
- [44] T. Kwon, "Privacy preservation with X.509 standard certificates," *Inf. Sci.*, vol. 181, no. 13, pp. 2906–2921, 2011.

- [45] J. Diaz, D. Arroyo, and F. B. Rodriguez, "New X.509-based mechanisms for fair anonymity management," *Comput. Security*, vol. 46, pp. 111–125, 2014.
- [46] J. Diaz, D. Arroyo, and F. B. Rodriguez, "Anonymity revocation through standard infrastructures," in *Proc. Eur. Public Key Infrastructure Workshop*, 2012, pp. 112–127.
- [47] A. Kiayias, Y. Tsiounis, and M. Yung, "Traceable signatures," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, pp. 571–589.
- [48] S. G. Choi, K. Park, and M. Yung, "Short traceable signatures based on bilinear pairings," in *Proc. Int. Workshop Security*, 2006, pp. 88–103.
- [49] L. Malina, J. Hajny, and Z. Martinasek, "Efficient group signatures with verifier-local revocation employing a natural expiration," in *Proc. Int. Conf. Security Cryptography*, 2013, pp. 555–560.
- [50] L. Malina, J. Hajny, and V. Zeman, "Light-weight group signatures with time-bound membership," *Security Commun. Netw.*, vol. 9, no. 7, pp. 599–612, 2016.
- [51] A. Miyaji, M. Nakabayashi, and S. Takano, "New explicit conditions of elliptic curve traces for FR-reduction," *Inst. Electron. Inf. Commun. Eng. Trans.*, vol. 84-A, no. 5, pp. 1234–1243, 2001.
- [52] T. Kim and R. Barbulescu, "Extended tower number field sieve: A new complexity for the medium prime case," in *Proc. Annu. Int. Cryptology Conf.*, 2016, pp. 543–571.
- [53] P. Sarkar and S. Singh, "A general polynomial selection method and new asymptotic complexities for the tower number field sieve algorithm," in *Proc. Adv. Cryptology 13th Int. Conf. Theory Appl. Cryptology Inf. Security*, 2016, pp. 37–62.
- [54] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *Proc. Int. Conf. Security Cryptography Netw.*, 2006, pp. 111–125.
- [55] N. Attrapadung, G. Hanaoka, K. Ogawa, G. Ohtake, H. Watanabe, and S. Yamada, "Attribute-based encryption for range attributes," in *Proc. Int. Conf. Security Cryptography Netw.*, 2016, pp. 42–61.
- [56] M. Green and S. Hohenberger, "Universally composable adaptive oblivious transfer," in *Proc. Adv. Cryptology 13th Int. Conf. Theory Appl. Cryptology Inf. Security*, 2008, pp. 179–197.
- [57] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous attestation using the strong Diffie Hellman assumption revisited," in *Proc. Int. Conf. Trust Trustworthy Comput.*, 2016, pp. 1–20.
- [58] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2003, pp. 614–629.
- [59] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Proc. Cryptographers Track. RSA Conf.*, 2005, pp. 136–153.
- [60] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth, "Foundations of fully dynamic group signatures," in *Proc. Int. Conf. Appl. Cryptography Netw. Security*, 2016, pp. 117–136.
- [61] Y. Sakai, J. C. N. Schuldt, K. Emura, G. Hanaoka, and K. Ohta, "On the security of dynamic group signatures: Preventing signature hijacking," in *Proc. Int. Workshop Public Key Cryptography*, 2012, pp. 715–732.
- [62] J. Camenisch and J. Groth, "Group signatures: Better efficiency and new theoretical aspects," in *Proc. Int. Conf. Security Commun. Netw.*, 2004, pp. 120–133.
- [63] M. Abdalla and B. Warinschi, "On the minimal assumptions of group signature schemes," in *Proc. Int. Conf. Inf. Commun. Security*, 2004, pp. 1–13.
- [64] K. Emura, G. Hanaoka, Y. Sakai, and J. C. N. Schuldt, "Group signature implies public-key encryption with non-interactive opening," *Int. J. Inf. Security*, vol. 13, no. 1, pp. 51–62, 2014.
- [65] G. Ohtake, A. Fujii, G. Hanaoka, and K. Ogawa, "On the theoretical gap between group signatures with and without unlinkability," in *Proc. Int. Conf. Cryptology Africa*, 2009, pp. 149–166.
- [66] D. Derler and D. Slamanig, "Fully-anonymous short dynamic group signatures without encryption," *Cryptology ePrint Archive*, Graz Univ. of Technol., Graz, Austria, Rep. 2016/154, 2016. [Online]. Available: <http://eprint.iacr.org/2016/154>
- [67] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Annu. Int. Cryptology Conf.*, 1986, pp. 186–194.
- [68] K. Kurosawa, "Multi-recipient public-key encryption with shortened ciphertext," in *Proc. Int. Workshop Public Key Cryptography*, 2002, pp. 48–63.
- [69] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proc. 23rd ACM SIG-SAC Conf. Comput. Commun. Security*, 2006, pp. 390–399.
- [70] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 1996, pp. 387–398.
- [71] OpenSSL Project, "OpenSSL: Cryptography and SSL/TLS Toolkit." [Online]. Available: <https://www.openssl.org>
- [72] M. Bellare and P. Rogaway, "The exact security of digital signatures - how to sign with RSA and Rabin," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 1996, pp. 399–416.
- [73] S. Micali and L. Reyzin, "Improving the exact security of digital signature schemes," *J. Cryptology*, vol. 15, no. 1, pp. 1–18, 2002.



Keita Emura received the ME degree from Kanazawa University, and the PhD degree in information science from the Japan Advanced Institute of Science and Technology (JAIST), in 2004 and 2010, where he was with the Center for Highly Dependable Embedded Systems Technology as a post-doctoral researcher, in 2010–2012. He was with Fujitsu Hokenriku Systems Ltd., from 2004 to 2006. He has been a researcher with the National Institute of Information and Communications Technology (NICT), since 2012, and has been a senior researcher at NICT since 2014. His research interests include public-key cryptography and information security. He was a recipient of the SCIS Innovation Paper Award from IEICE in 2012, the CSS Best Paper Award from IPSJ in 2016, and the IPSJ Yamashita SIG Research Award in 2017. He is a member of the IEICE, IPSJ, and the IACR.



Takuya Hayashi received the bachelor of media architecture and the master of systems information science degrees from the Future University-Hakodate, in 2008 and 2010, respectively, and the doctor of functional mathematics degree from Kyushu University, in 2013. He is currently an assistant professor with Kobe University, and an invited advisor in the National Institute of Information and Communications Technology. His current research interests include cryptanalysis and efficient implementation of public key cryptosystems. He was awarded SCIS paper prize from IEICE in 2010, and DOCOMO Mobile Science Award in 2013.



Ai Ishida received the BS and MS degrees from the Tokyo Institute of Technology, in 2013 and 2015, respectively. She is now working toward the third grade PhD degree in the Tokyo Institute of Technology. She received LA/EATCS Best Presentation Award in 2015, the IWSEC Best Poster Award in 2015, and SCIS Paper Prize from IEICE in 2015.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

I Sense You by Breath: Speaker Recognition via Breath Biometrics

Li Lu¹, Member, IEEE, Lingshuang Liu, Muhammad Jawad Hussain¹, and Yongshuai Liu

Abstract—Over last two decades, Speaker Recognition has primarily been focused on *source*, *system*, and *prosodic* features of the speech. The breath, however, has either been treated as a trivial part of the speech, or considered a noise entity. Our observation reveals that breath is a unique fingerprint of human respiratory system which offers overwhelming results for Speaker Recognition. Moreover, its passive nature, short-duration, fewer occurrences and simple processing results to a light-weight, text-independent and transparent system, which we articulate as BreathID. The breath features are extracted and classified by Mel Frequency Cepstral Coefficients, MFCC, based template matching technique. The verification is performed by a similarity based scheme, whose efficiency competes with classification algorithms. We process a data set collected from 50 users. Our system offers a 0.04 percent False Identification Rate, FIR, for Speaker Identification, and 0.12 percent False Acceptance Rate, FAR, and 0.15 percent False Rejection Rate, FRR, for Speaker Verification. We further evaluate our scheme under various practical modalities, like text in-dependence, replay scenario, users' motion status (sitting and walking), recording equipment (03 smartphones and 02 microphones), recording period (08 months), and bilingual contents (English and Chinese). Though we use Matlab to formulate a fine-grained approach, we foresee breath biometric as a viable security measure for practical realizations.

Index Terms—Breath biometrics, speaker recognition, MFCC

1 INTRODUCTION

SPEAKER Recognition is a generic term used for two problems. In *Speaker Identification*, the identity of a person is ascertained while during *Speaker Verification*, the claimed identity is verified. This is realized through Training phase which computes the reference model from feature vectors, and Testing phase which finds the similarity score. The speech contents are categorized as *text-dependent* (only specific words), *text-independent* (no restriction of words) and *text-prompted* (speak any of the specific words). Though speech biometrics savour the degradation (microphone, channel), health (sickness, emotions) and iteration (mimicking), recent advancements in speech processing have resulted in its real-time and truly functional applications [1], [2], [3], [4], [5], [6].

Our research stems from the observation that existing Speaker Recognition schemes extract “feature vectors” from three underlying stimuli, based upon *Source-Filter* and *Source-System* models, separately or in combination [7], [8], [9], [10], [11], [12], [13]. The *excitation source* information is represented through linear prediction residual samples in shape of glottal pulse parameters. The *vocal tract* information is acquired from cepstral coefficients. The *prosodic* information is sought through statistics and temporal dynamics of duration, pitch, and energy. However, the aerodynamic breath process is foretold as a mere energy source for sound generation, and

processed as an integral part of the speech. The existing studies on breath have been focused on its detection and removal for improving sound quality [14], [15], speech-to-text algorithms [16], [17], training transcriptionists [18] and psychological diagnosis [19], [20]. Besides, breath is chemically analyzed for health diagnosis [21]. For the first time, we exploit the uniqueness of breath features and propose BreathID for Speaker Recognition.

Breath is an anatomical fingerprint of respiratory system due to intrapulmonary pressure and governed through vocal tract, lungs, trachea, diaphragm and respiratory muscles [22], [23]. The non-instantaneous airflow results to silence periods (≥ 20 msec) at both ends of a breath (Fig. 1). Compared with speech, breath is weaker in energy, shorter in time (100-400 msec), has low occurrences (12-18 per min), and overlaps at low frequencies (100 Hz-1 kHz). Another vulnerability arises as it closely resembles phonemes and fricative consonants, like /tʃ/ in church and <3> in vision. The main challenge we face in BreathID is the demarcation and processing of breath itself. Our scheme comprises of Breath Demarcation, Feature Extraction and Feature Matching stages, as explained below:

1) *Accurate Breath Demarcation*. We observe three appealing works for breath extraction in professional audio systems [15], [18], [24]. Our algorithm is inspired by [15] due to its simplicity and high accuracy. It is used for real-time extraction of breath in high quality speech and song recordings, and comprises of three phases: First, a generic breath template is computed from MFCC; Second, breath is detected using a template matching technique which is derived from two-class Gaussian classifier. The similarity is calculated from Zero Crossing Rate (ZCR), Short Time Energy (STE),

• The authors are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Sichuan Sheng 610051, China. E-mail: luli2009@uestc.edu.cn, {lslui, liuys}@std.uestc.edu.cn, j19197@gmail.com.

Manuscript received 30 Sept. 2016; revised 18 Apr. 2017; accepted 19 Oct. 2017. Date of publication 30 Oct. 2017; date of current version 18 Mar. 2020. (Corresponding author: Li Lu.)

Digital Object Identifier no. 10.1109/TDSC.2017.2767587

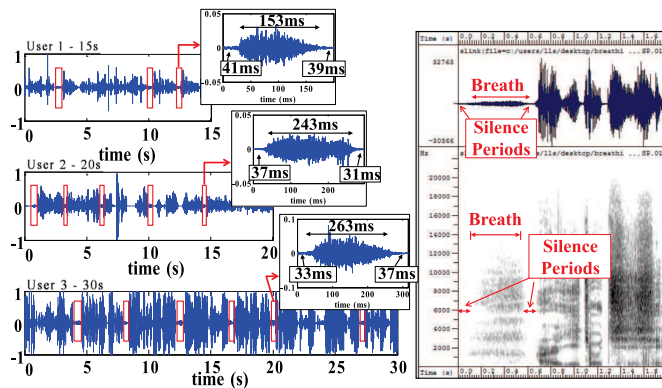


Fig. 1. Breath segments in 15, 20 and 30 sec speech of three individuals (left), and spectrogram of an arbitrary breath signal (right).

and $B(X_i, T, V, S)$ index where T, V, S are cepstrogram, variance and SVD (Singular Value Decomposition) matrices for segment X_i ; Third, false positives are removed by breath duration thresholds, energy thresholds, ZCR, Spectral Slope, and Edge Detection techniques.

2) *Feature Extraction*. In speech processing, typically used schemes are MFCC, Real Cepstral Coefficients (RCC), Linear Predictive Coding (LPC), Perceptual Linear Predictive Cepstral Coefficients (PLP), Principal Component analysis (PCA), Linear Discriminant analysis (LDA), Independent Component Analysis (ICA) and Perceptual Log Area Ratio (PLAR). Amongst these, MFCC suits us more because Mel filters can effectively capture the low frequency breath contents. With an aim of low complexity and high accuracy, we first consider MFCC, STE, ZCR, Spectral Slope, Fundamental Frequency (F_0) and Formant (F_1) as potential candidates in BreathID. We finally select MFCC feature vectors because of their superior results.

3) *Feature Matching*. The typically used pattern matching or classification algorithms are (Gaussian Mixture Model (GMM), Vector Quantization (VQ), Hidden Markov Model (HMM), Dynamic Time Warping (DTW), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP) and Artificial Neural Network (ANN). In BreathID, we evaluate GMM, HMM, SVM, ANN and KNN (K-Nearest Neighbor), whereby GMM and HMM outperform both in profiling and verification phases. In addition, we formulate a simple similarity based scheme, *Decision Maker*, which uses mean and variance matrices along with basic vector operations, and outperforms classification algorithms in our case.

We first evaluate BreathID algorithm while carrying out a measurement-based study on 50 volunteers with age ranging from 10-60 years. A total of 25 recordings were taken from each volunteer over a span of one month, whereby each recording consisted of 02 mins. The evaluation was aimed for two applications: User Identification and User Verification. Our system can effectively recognize users with 0.04 percent FIR, and offers < 0.12 percent FAR and 0.15 percent FRR for Verification with breath segments in 20 sec of speech. The false rates almost reduce to zero for 60 sec audio in Profiling and 20 sec in Verification. The prime limitation is the required length of the recorded voice. We also examine FAR and FRR in relation to length of the recorded speech and number of breath segments. Our

algorithm is devised in Matlab that can verify a user within 260 msec if we use 60 sec speech during Profiling, and 20 sec during Verification.

Next, we evaluate BreathID for various practical modalities, including: *text in-dependence*, *audio replay*, *recording equipment*, *motion status*, *recording period* and *language*. This involves new voice recordings from 20 volunteers. The *text-independence* was checked from 04 text contents, for which Equal Error Rate (EER) remains below 0.15 percent. For *audio replay*, the recorded voice was transmitted through speaker and then re-recorded. The average EER came out to be 44.75 percent due to degradation by speaker-air-microphone channel. A negligible influence of *recoding equipment* was noticed amongst 03 smartphones and 02 commercial microphones. For *motion status*, we observe a maximum EER of 0.14 percent once the speaker is sitting and 0.15 percent during the walk. For *time consistency*, the recordings were performed once per week over a period of 08 months. The results show a maximum EER of 0.16 percent. For *Multilingualism*, we observe a maximum EER of 0.17 percent for both English and Chinese (Mandarin) languages.

Because of short duration, fewer occurrence, and simple processing, the overall system becomes computationally light-weight and economical in terms of processing time. The system also becomes memory efficient as a single breath profile merely occupies 423 KBytes. We foresee BreathID as a passive and transparent system because it can extract breath features as long as the user is speaking, regardless of language or text contents, and any feedback from the user.

The remainder of the paper is organized as follows: Section 2 discusses the uniqueness of breath biometrics. An in-depth discussion on three phases of BreathID is followed in Section 4. Section 4.4 discusses the evaluation setup and parameters considered. Section 5 presents the evaluation results. A brief overview of Biometric-based User Recognition is presented in Section 3 including MFCC based schemes. Paper concludes at Section 6.

2 UNIQUENESS OF BREATH BIOMETRICS

The central question we answer here is why breath has remained dormant in Speaker Recognition and processed as integral part of the speech, or foretold as respiration noise? We first present a brief overview of Source-Filter and Source/System models, which are the underlying basics of speech synthesis and analysis. This is followed by comparison of breath with speech.

The *source-filter theory* considers speech as the response of vocal-tract system, and gives a good approximation of non-linear and time-variant speech sounds [8], [25]. The “source” refers to four sources of speech sounds: aspiration, frication, glottal (or phonation) and transient sources. The vocal-tract acts like a “filter” which takes input from either or combination of these sources, and its filter response results to vowels, consonants or any speech sound [25], [26], [27]. This is the generic idea while further details govern the generation of pitch, voice quality, harmonics, resonance characteristics, radiation response, etc.

In *source/system model*, the speech is modeled by a linear and slowly varying discrete time system which is either excited by the random noise during unvoiced speech, or quasi-periodic pulses during voiced speech [9], [12], [13],

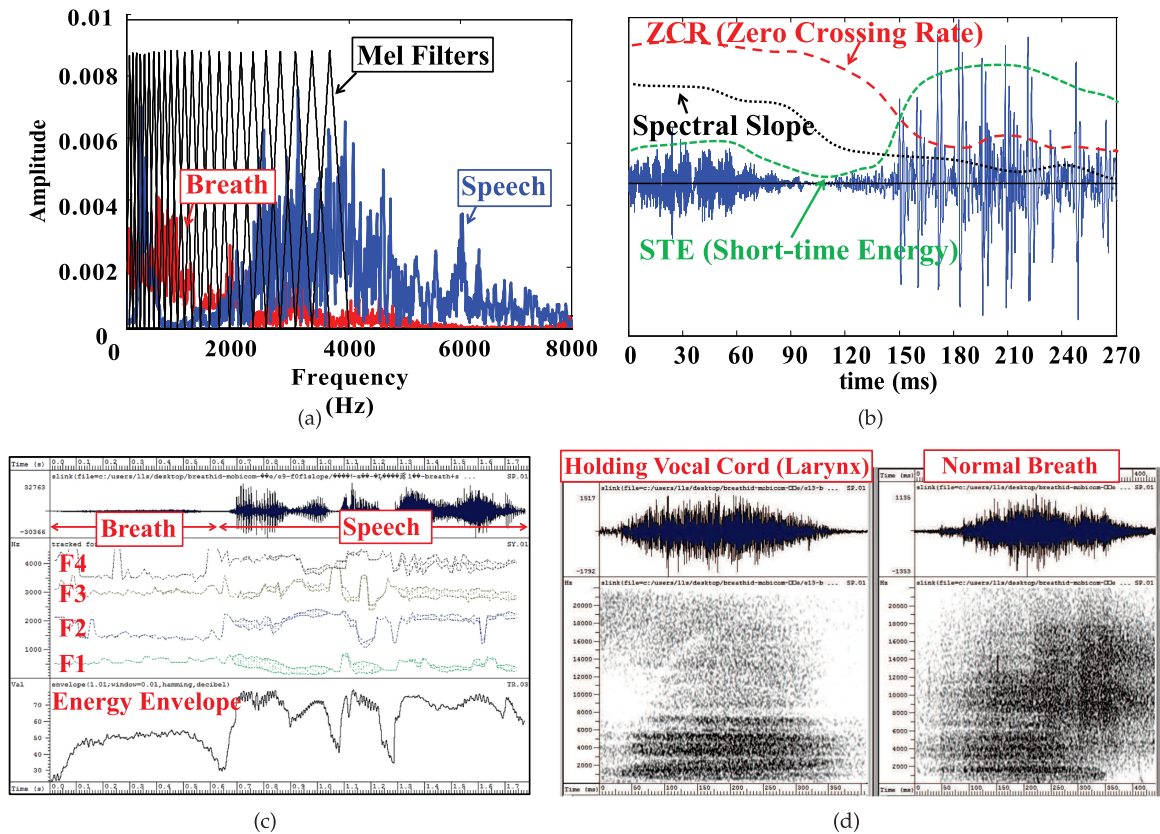


Fig. 2. Comparison of breath with speech signals. (a) mel filters; (b) ZCR, slope, STE; (c) formants; (d) normal and unusual breath.

[28], [29]. The *source* contains pitch and voicing characteristics which are much prone to errors [30]. Therefore, it is either rarely used in Speaker Recognition or augmented with other features. The *system* corresponds to smooth envelope of power spectrum which is normally computed through linear prediction or Mel filter analysis. This is widely used in Speaker Recognition systems in shape of cepstral coefficients.

We observe that both models treat breath as an integral part of *source*, which is transformed into speech during voiced speech, or, noise during unvoiced speech. In fact, respiration (i.e., breathing) is considered as a *power mechanism* to provide energy for sound [31], [32], [33]. Moreover, during “Respiration for Speech”, the breathing is controlled (exhalation is longer than inhalation), while during “Respiration for Life”, both phases are nearly the same [34].

We foresee breath as a physical fingerprint of whole respiratory system that includes lungs, diaphragm, intercostal muscles, and air passageway including bronchi, trachea, larynx and vocal tract and mouth cavity [22], [23], [35]. Breathing is governed through intrapulmonary pressure and air flow direction, and controlled by muscular movements. In inspiration, the respiratory muscles contract which results in decrease of pressure, and makes the air to enter the lungs. Similarly, the expiration decreases the intrapulmonary volume which increases the pressure and expels the air [36], [37]. Because of this anatomy, a certain pause exists between breath and the preceding and following utterances. Typically, breath lasts between 100-400 msec depending upon age and gender. The silence periods reside over 20 msec which significantly demarcate a breath event.

For illustration, Fig. 2 compares a breath segment with a speech signal to show the difference using the speech processing tools we used in BreathID. Fig. 2a shows both signal at the output of Mel Filter bank (depicted by rectangular lines). Fig. 2b shows the ZCR, Spectral Slope and STE. The noise-like breath segment has comparatively high ZCR and low STE, as perceived. In both Figs. 2a and 2b, we choose a relatively low-energy speech signal to emphasize the characteristics of breath, otherwise, most of the recorded breaths were quite smaller in amplitude compared to the speech signal, as in Fig. 2c. Next, Fig. 2c shows the Energy Envelope and the formants (F_1 , F_2 , F_3 and F_4) which are discussed in Section 4.2.1.

Lastly, we highlight that a person can deliberately or accidentally alter his breath signature if respiratory organs are disturbed. Fig. 2d shows the spectrogram during normal breathing, compared with once the speaker holds his vocal cords (larynx). Another contradiction arises during respiratory dysfunction or irregular breathing. Like high metabolic demand (e.g., exercise) or asthma. In BreathID, we suppose an honest candidate who breaths under normal health conditions.

3 RELATE WORK

The underlying principle of Biometric-based User Recognition focuses on “who you are” which differs from Conventional User Recognition approach that mainly relies on “what you have” or “what you know”. The *physiological biometrics* are physical characteristics measured at some particular time like fingerprint [38], retina or iris pattern [39], and face or hand geometry [40]. Though these schemes are widely adopted by Access control systems,

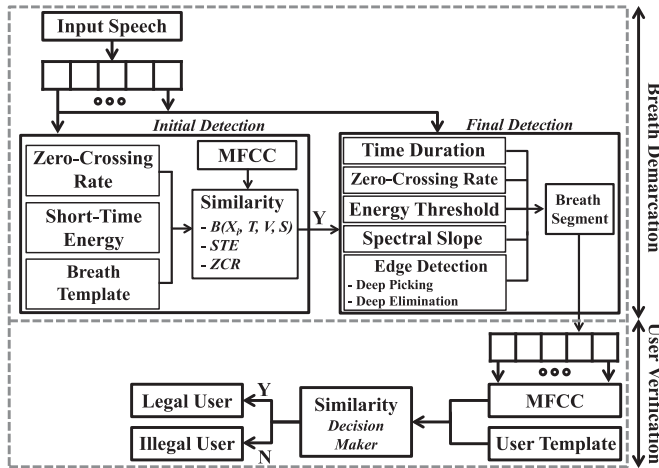


Fig. 3. System block diagram.

they require specialized hardware like fingerprint identifier, HD camera etc.

The *behavioural biometrics* consist of actions that are learned or acquired over time and can be deliberately changed, like signature, voice or gait. On PCs, key and pressure dynamics [41], [42], and on-screen mouse movements [43], [44] has been exploited for User Recognition. Recently, a large number of User Recognition methods on mobile devices have been proposed [45], [46], [47], [48], [49], [50] which recognize the users by exploiting their behavior while using the device.

Speech-based User Recognition is also called Speaker Recognition. From the aspect of speech content, two main schemes are text-dependent and text-independent. The former uses the same text for training and testing user models and mainly utilizes GMM, VQ, ANN and SVM for speech modeling [51], [52], [53], [54]. Current results show a recognition accuracy from 70 to 98 percent and False Rate from 4 to 12 percent for User Identification with MFCC and ANN classifier [51]. The latter approach can verify a user's speeches even if the user speaks different texts for model training and testing. Current schemes [55], [56], [57] typically employ MFCC to extract features vectors, along with classification algorithms for making user profiles. The achieved accuracy is around 96 percent for PC [58] and mobile platforms [55]. The chief drawback here is the large amount of speech data required for model profiling, and use of complex classifiers. In addition, the accuracy can be degraded by ambient channel noise and the recording equipment.

Within the realm of speech processing, MFCC is acclaimed over other techniques [59], and its many variants have been investigated. Delta MelFCC (DMFCC) explores second order derivatives of MFCC for intra-speaker variability [58]. Inverse-MFCC (IMFCC) reverses the high density filters to higher frequency contents [60]. Modified MFCC (MMFCC) uses an auditory model for Automatic Speaker Recognition (ASR) [61]. Multi-dimensional MFCC has been used for spotting keywords [62]. The main weakness of MFCC is its low noise resilience [63], [64], for which SMN-CMN-MFCC (Spectrum Mean Normalization, SMN, Cepstral Mean Normalization, CMN) is proposed to suppress additive and convolutional noise [65]. A MFCC division-based scheme is proposed in [64] for noise robust

Speaker Identification. MFCC is modified with MMSE (minimum mean square error) algorithm for noise robustness [63]. Different variants of MFCC have been evaluated in [66] for VoIP Networks. Different windowing functions in MFCC have been investigated for Monolingual and Crosslingual Speaker Identification in [67].

Many schemes combine MFCC with other techniques for improved results. MFCC has been combined with Wavelets for music classification [68]; with phase [69] and, phase and slope for Speaker Identification [70]; MFCC with prosodic features for language identification [71]; MFCC and IMFCC combined for Speaker Identification [60]; complementary filter bank is fused for a text-independent Speaker Identification [72]; MFCC with Wiener Filter is proposed for noise robust Speaker Identification [73], amongst others [74], [75], [76], [77].

4 SYSTEM ARCHITECTURE

BreathID consists of three modules: Breath Demarcation, Feature Extraction and Feature Matching (Decision Maker), as shown in Fig. 3. Breath Demarcation makes a generic breath template to extract the fine-grained breath segments, and filters the overlapping contents of speech. Feature Extraction calculates the breath features vectors while preserving the identity contents of the speakers. Decision Maker finds the similarity score by comparing the features vectors of known speech signals with the reference model.

4.1 Breath Demarcation

The accurate detection and extraction of breath involves three steps: Breath Template Construction; Initial Detection Phase; Final Detection Phase. As our Breath Demarcation scheme is inspired by Breath extraction algorithm in [15], we briefly explain each of three steps as per their utilization in BreathID.

4.1.1 Breath Template Construction

The breath template is constructed using several breath samples. Such samples can be arbitrary breath events which can be derived from one or more speakers. As per our experiences, a template made from 50 arbitrary breath segments (voice sample of 3-4 mins) offers a breath demarcation accuracy of 99.6 percent, which is the procedure we follow in our paper. The template is only used to accurately and exactly demarcate breath event, and is independent of User Recognition system.

The template construction has three main goals: template should serve as a generic prototype which can extract breath segments from any unknown voice signal; it should contain the information which can distinguish a breath segment from other phonemes and consonants; it should be compact for computational efficiency. For brevity, we explain template generation in four steps.

Step 1: Input voice is converted into isolated example sets, each with 100 msec frame length. Each frame is further divided into short consecutive subframes of 10 msec (5 msec overlap hop). The choice of frame length and hop period would affect the breath extraction step and computational time, the details of which can be referenced from [15], [78]. The subframes are next pre-emphasized using a first order difference filter ($H(z) = 1 - \alpha \cdot z^{-1}$, where $\alpha \approx 0.095$ [15]).

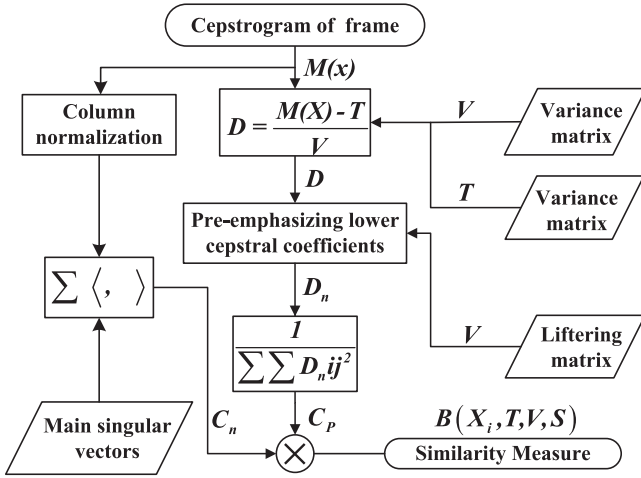


Fig. 4. Initial detection phase.

Step 2: For each subframe, MFCC is computed to give a short-time cepstragram matrix, whose columns are the MFCC vectors. The constant values are then removed for each column (DC removal). Such template construction fulfills the stipulated goals with small number of parameters [79], [80]. Moreover, it has been shown to effectively recognize phonemes and low frequency speech contents with high resolution [17], [81], [82].

Step 3: Mean cepstragram is computed by averaging the matrices of example set. This results to breath template matrix T given as $T = \frac{1}{N} \sum_{i=1}^N (M(X_i))$, where N is the number of samples in the example set, and $M(X_i)$ is the MFCC vector matrix of each subframe.

Step 4: Similarly, a variance matrix V is computed as follows: the matrices of example set are concatenated into one matrix, and SVD (Singular Value Decomposition) of the resulting matrix is computed. Then, we get the normalized singular vector, S , corresponding to the largest singular value. We use SVD transform because of its information packing property [83]. The T and V matrices are used together as a breath template for initial breath detection.

4.1.2 Initial Detection Phase

Each short-time frame is compared to breath template and marked as breathy or nonbreathy. It involves a template matching technique with scaled euclidean distances, which becomes a special case of two-class Gaussian Classifier with a diagonal covariance matrix [84]. The underlying reason is the computation of discrete cosine transform at the last step of MFCC, which is used to decorrelate the Mel-scale filter log-energies. As shown in Fig. 4, the similarity measure, $B(X_i, T, V, S)$, is based upon C_p and C_n scores, as explained below:

Step 1: Compute the normalized difference matrix, $D = \frac{M(X) - T}{V}$. The element-by-element normalization by V compensates the difference in distribution of various cepstral coefficients.

Step 2: The difference matrix is *liftered* by multiplying each column with a half-Hamming window. Such operation emphasizes the lower cepstral coefficients, which has been observed to achieve better separation between breath and other sounds [15], [79].

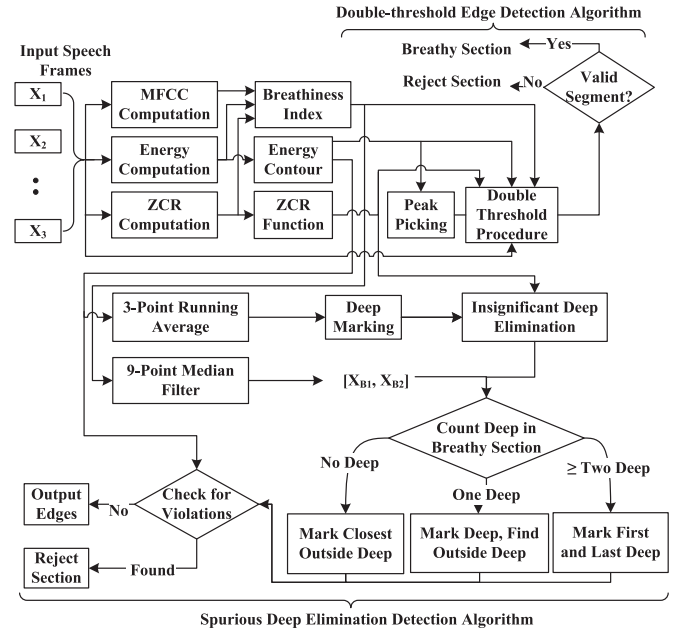


Fig. 5. Final detection phase.

Step 3: The first similarity measure, C_p , is calculated as $C_p = 1 / \sum_{i=1}^n \sum_{j=1}^{N_c} |D_{ij}|^2$, where n represents the number of subframes, and N_c is the number of MFCC for each subframe. The value of C_p will be high once the cepstragram is very similar to the template (elements of difference matrix are small).

Step 4: The second similarity measure, C_n , is computed by taking the inner product of singular vector S and the normalized columns of the cepstragram. The result will be small in case breath template is correlated with other phonemes. The $B(X_i, T, V, S)$ is then given as the product of two parameters, i.e., $C_p \cdot C_n$.

Step 5: An event can be classified as breathy once three conditions hold: $B(X_i, T, V, S)$ is greater than threshold $B_m/2$, where B_m is minimum value of the similarity measures between each of the example sets and template; the STE is below the average energy of voiced speech in example sets; the ZCR is below 0.25 (for 44 kHz sampling rate) [15].

Initial Detection Phase correlates a breath sample to reference template, and the result is a binary breathiness index. Though this phase shows high detection sensitivity, it is still susceptible to false positive detections. Importantly, its time resolution is not very accurate to detect the exact beginning and end of the breath events. These two vulnerabilities are addressed by Final Detection Phase.

4.1.3 Final Detection Phase

It considers neighbouring frames to solve the problem of false positives and inaccurate edge detection by using various time and frequency domain parameters. Fig. 5 shows the two inputs to this stage: First, the time-domain speech segment, which was also fed to Initial Detection Phase to check the presence of breath within current speech segment. Second, the binary breathiness index (output of Initial Detection) which tells whether the current speech signal has any breath segment or not.

For *False Detection Elimination*, various thresholds are applied. These include Preliminary Duration Threshold, Upper Energy Threshold, Lower ZCR Threshold, Upper ZCR Threshold, and Final Duration Threshold. The key concept is to exploit the difference in threshold values for breath, compared to false signals. An in-depth discussion can be found in [15].

The *Accurate Edge Detection* is designed according to the nature of a breath signal, i.e., a genuine breath signal is expected to have a peak in the local energy function, which is accompanied by two noticeable deeps on each end of the peak (i.e., silence periods). To this end, two methods are applied. First uses the double energy threshold and compares the energy of breath's peak with silence periods. It is termed as *Double Energy Threshold and Deep Picking* method. However, it become less accurate once a short noise (or unvoiced speech signals) appear. To solve this, the second method is used to eliminate the spurious peak and deep points. This is termed as *Edge Marking With Spurious Deep Elimination* methods. Further details for both methods can be referenced in [15].

4.2 Feature Extraction

The general methodology of Speaker Recognition involves extracting discriminatory and low-dimensional features which are suitable for statistical modeling, distance computation or similarity measure. Since the speech of an individual will have similar but still differently arranges feature vectors, this step is mostly followed by Feature Classification step. A classifier transforms these feature vectors to a reference Speaker Model. For BreathID, both steps are explained below:

4.2.1 Extraction of Feature Vectors

Based upon our experiences with breath, we consider the basic time, frequency and energy based operations to process raw breath signals. These include MFCC, STE, ZCR, Spectral Slope, F_0 and F_1 .

- **MFCC.** Mel-Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound [80]. Cepstral (nonlinear "spectrum-of-a-spectrum") coefficients obtained for Mel spectrum are referred to as Mel-Frequency Cepstral Coefficients [85]. The human ear does not perceive frequencies in a linear fashion, it acts like an unevenly spaced filter bank, with higher number of filters in the low frequency area and vice versa. For example, doubling a frequency from 1 to 2 KHz, does not double the pitch. To address, Mel Scale was derived in 1937 [86] which maps the audio signals into these non-linear perceptions of it.

MFCC computation starts with Fast Fourier Transform (FFT) of input sampled audio to calculate the complex sinusoids coefficients (performs Discrete Fourier Transform, DFT). The output is passed through a filterbank based upon Mel scale, given as $mel(x) = 1125 \log(1 + x/700)$ [87]. The triangular bandpass filters are 50 percent overlapped, follow approximately a linear scale from 0 to 1000 Hz, and a logarithmic scale at higher frequencies. For an input signal with N samples, the filters are given by [87]

$$f_m = \left(\frac{N}{F_s} \right) mel^{-1} \left(mel(f_l) + m \frac{mel(f_h) - mel(f_l)}{M + 1} \right) \quad (1)$$

$$0 \leq m \leq M + 1.$$

where M is total number of filters, F_s is the sampling frequency, f_l and f_h are the lowest and highest frequencies of the filter f_m . The normalized triangular filters, $H_m(k)$ are then given by [88]

$$H_m(k) = \begin{cases} 0 & k < f_{m-1} \\ \frac{2(k-f_{m-1})}{(f_{m+1}-f_{m-1})(f_m-f_{m-1})} & f_{m-1} \leq k \leq f_m \\ \frac{2(f_{m+1}-k)}{(f_{m+1}-f_m)(f_m-f_{m-1})} & f_m < k \leq f_{m+1} \\ 0 & k > f_{m+1} \end{cases} \quad (2)$$

$$1 \leq m \leq M.$$

Next, the log output of filterbank is given by [89]

$$S_m = \log \left[\sum_{k=0}^{N-1} |X_k|^2 H_m(k) \right] \quad 1 \leq m \leq M. \quad (3)$$

MFCC is the DCT (Discrete Cosine Transform) of the M filter outputs, as [90]

$$c_n = \sum_{m=0}^{M-1} S_m \cos \left(\pi n \left(m - \frac{1}{2} \right) / M \right) \quad 0 \leq n \leq M. \quad (4)$$

This filtering action compacts the information while reducing the number of coefficients. The sample averaging also reduces the variance of DFT within each filter. Finally, a logarithmic compression is performed followed by DCT to the energy vectors. DCT primarily serves two purposes: First, it separates the slow varying spectral envelope (like vocal tract) contents from fast varying speech excitation signals. This way, MFCC only retains the low-order coefficients related to vocal tract. Second, it decorrelates the elements of the feature vector as the elements of log filterbank vector exhibit correlation because of spectral characteristics of speech and overlapping nature of the filters. The resulting decorrelated coefficients are suitable for further analysis.

- **Short-Time Energy (STE)** is an indication of signal amplitude in the interval around a specific signal point. It is a time-domain analysis tool which is primarily used to differentiate voiced and unvoiced sounds in speech from silence. The STE can be computed as [91]

$$E = \frac{1}{N} \sum_{n=N_0}^{N_0+N-1} x^2[n], \quad (5)$$

where $x[n]$ is the sampled audio signal, and N is the window length in samples (10 msec in BreathID).

- **Zero-Crossing Rate (ZCR)** is the number of times that breath waveform changes its sign. By analogy, breath has higher ZCR than voiced speech. However, our experimental results show that ZCR of unvoiced phonemes, such as fricative consonants, is

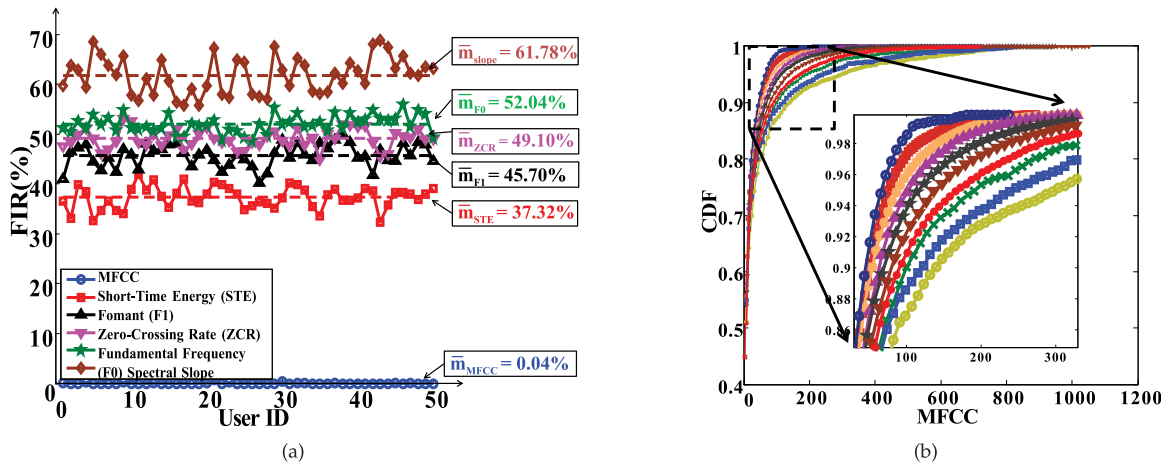


Fig. 6. Feature extraction phase. The feature vectors are based upon MFCC values which outperform the results of ZCR, STE, spectral slope, F0 and F1 for false identification rate. (a) FIR of MFCC with other features; (b) CDF of MFCC for ten speakers.

usually higher than the breath. In BreathID, we normalize ZCR by the window length N (10 msec), as given by [15], [91]

$$ZCR = \frac{1}{N} \sum_{n=N_0+1}^{N_0+N-1} 0.5 |sgn(x[n]) - sgn(x[n-1])|. \quad (6)$$

- Spectral Slope is a measure of voice quality which is reflected in intensity of the harmonics, and more generally, in the shape of power spectrum. We observe a significant difference in slope steepness between breath and silence/speech signals. For slope in breath is usually steep in the middle and flat at both edges. We compute this by taking the DFT of the analysis subframe, evaluating its magnitude at frequencies of $\pi/2$ and π (corresponding to 11 and 22 kHz), and computing the slope of the straight line fit between these two points. In contrary, the voiced speech has most of spectral energy in lower frequencies (typically below 4 KHz) [83], and its spectrum is expected to be rather flat between 11-22 KHz [15].
- Fundamental Frequency (F_0) is defined as the lowest frequency of the breath signals. Though we foresee breath as a noise-like signal, we consider F_0 because it is less sensitive to channel distortions and additive noise, once compared with spectral or cepstral features [92].
- Formant (F_1) is measured as amplitude peaks in the frequency spectrum of the sound, and gives an estimate of the vocal tract resonance [93]. The commonly used Klatt Synthesizer [94] consists of parallel and series filters. Both filters model the spectral envelope of input speech, and their resonance frequencies are tuned to formant frequencies of the phonemes. We use parallel model because breath signals are similar to random noise-like signals for fricatives, while series model is appropriate for the modeling of fricatives and stops in voiced signals [10]. In BreathID, we use the lowest frequency formant, F_1 .

Next, we investigate whether these features are suitable for modeling users' breath or not. We conduct an empirical study in terms of the relationship between the accuracy of recognition with these features, and find out that merely the

MFCC feature vectors can provide accurate Speaker Recognition. This is evident from the FIR in Fig. 6 for all 50 users. The mean FIR for MFCC (denoted by \bar{m}_{MFCC}) is 0.04 percent while the closest is the STE ($\bar{m}_{STE} = 37.32$ percent). Fig. 6a shows the CDF of MFCC features for ten users. Though we can formulate a set of feature vectors based upon two or multiple parameters, we focus only on MFCC feature vectors as our system is the first proof-of-concept evaluation of Breath biometrics. Further optimizations can be followed as future work to our research, like MFCC combined with other parameters, or, optimizations within MFCC itself as discussed in Section 3.

4.2.2 Classification of Feature Vectors

Though feature vectors exhibit the identity contents of an individual, still they are differently arranged and need to be classified or matched to form a user's model. In speech processing, the feature extraction step is typically followed by classification step, for which various Pattern Matching and Classification algorithms have extensively been studied [95], [96], [97], [98], [99].

In BreathID, though classification can be performed by machine learning approaches, we find that feature vectors directly derived from MFCC can achieve a high level of recognition accuracy. Figs. 7a and 7b shows the FAR and FRR for Profiling (Training) stage for GMM, HMM, SVM, ANN, KNN algorithms, and MFCC (shown as BreathID). Amongst classifiers, GMM outperforms other algorithms similar to speech processing [100], [101], [102]. However, the results of MFCC feature vectors are considerably improved over GMM. The results become appealing once we consider time of speech signal required to achieve low false rates, i.e., MFCC reduces the FIR nearly to zero for 60 sec speech (12-18 breaths). Whereas, other algorithms still retain noticeable false rates even for 120 sec of speech. Importantly, MFCC feature set serves two purposes: First, we can formulate a simple and computationally efficient, yet very accurate and reliable system. In contrast, classification algorithms require complex models, necessitate multiple parameters and assumption, and resultantly cost higher time and space complexity. Second, MFCC based procedure is very fast, and achieves high accuracy even with very few training samples.

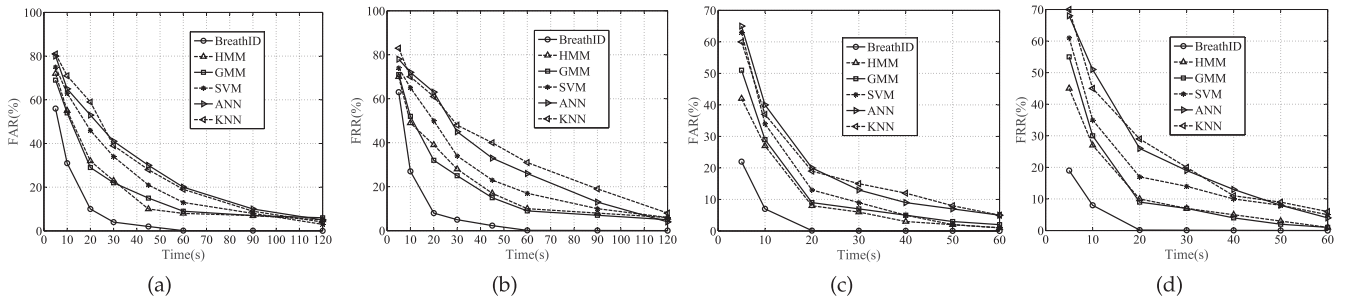


Fig. 7. Verification accuracy vs time in profiling and verification (training and testing phases). (a) FAR for profiling; (b) FRR for profiling; (c) FAR for verification; (d) FRR for verification.

4.3 Decision Maker

During verification step, breath samples from unknown speaker are matched to stored and reference models, and a similarity score is calculated. In analogy to training phase, we evaluate GMM, HMM, SVM, ANN and KNN algorithms, and a simple similarity based scheme (we term it as “Decision Maker”, labelled as BreathID in Figs. 7c and 7d). We observe promising results for Decision Maker as the false rates in Figs. 7c and 7d are reduced to decimals for 20 sec of voice input, whereas GMM and HMM achieve comparable accuracy over 60 sec of the speech. Decision Maker is devised from basic operations which simplifies the overall design in terms of computation, memory and time overhead. The process is explained below:

Step 1: Get the MFCC feature vectors of the valid (and registered) user, denoted as (a_1, a_2, \dots, a_n) , and then compute the average M and standard variance V of feature vector (a_1, a_2, \dots, a_n) .

Step 2: For the unknown user, calculate the MFCC feature vectors, denoted as (b_1, b_2, \dots, b_n) .

Step 3: Normalize the similarity of every element in (a_1, a_2, \dots, a_n) as $S_{a_k} = 1 / \sum_{i=1}^r \sum_{j=1}^c |(D_{a_k})_{ij}|^2$, where $D_{a_k} = \frac{a_k - M}{V}$, besides, r and c represents the number of rows and columns of D_{a_k} respectively.

Step 4: Sort the $(S_{a_1}, S_{a_2}, \dots, S_{a_n})$ in ascending order, denoted as (S_1, S_2, \dots, S_n) .

Step 5: Similarly, calculate the similarity of elements in (b_1, b_2, \dots, b_n) as $S_{b_k} = 1 / \sum_{i=1}^r \sum_{j=1}^c |(D_{b_k})_{ij}|^2$, where $D_{b_k} = \frac{b_k - M}{V}$, besides, r and c represents the number of rows and columns of D_{b_k} respectively.

Step 6: The position of S_{b_k} in sequential vector (S_1, S_2, \dots, S_n) is the similar proportion P_k of the element b_k . The average of P_k is the similarity between the measured user and the target user for feature F .

By comparing the similarities of all features with predefined thresholds, we can validate the measured user. The thresholds can be found through experimental measurements (see Section 4.4).

4.4 Performance Evaluation

Though we consider BreathID as an observation-based research for Speaker Recognition, we adopted a comprehensive methodology to evaluate its accuracy, efficiency and robustness under various practical scenarios. This sections elaborates the evaluation setup and the parameters considered.

4.5 Evaluation Setup

We collected data from 50 smartphone users from different genders and age groups. The voice is recorded through an Android application in a transparent and passive way. This means that our application runs in the background while the user can operate the smartphone in a natural and routine manner, like making an audio/video call. The App was installed on smartphones of 50 volunteers in our campus. Amongst these, nearly half were male and half were female. The age spans from 10 to 60 years (30 users within 10-30 years, 15 within 30-40 years, and 5 users from 40-60 years).

For evaluation of BreathID, each recording comprised of 02 mins while a total of 25 recordings were performed for each user during different time frames, over a period of one month. The generic breath template (to extract the breath only) is constructed from randomly selected voice samples to get 50 breath segments (Section 4.1.1). Each user breath profile was constructed from a single voice recording of 02 mins (Figs. 7a and 7b), while remaining 24 recordings were used for analysis. Fig. 8 shows the small variations between FAR and FRR for user verification. Each value shows the max/min span of 24 recordings for 50 users, while we consider the mean value during our discussion.

The recordings for various practical modalities were carried out separately on 20 volunteers. To include the degradation of recording device, we considered 03 smartphones and 02 commercial microphones, as the quality of microphone can affect spectral slope, noise level, influence of room acoustics etc [103]. The other scenarios, including multilingualism, motion status, text-independence and

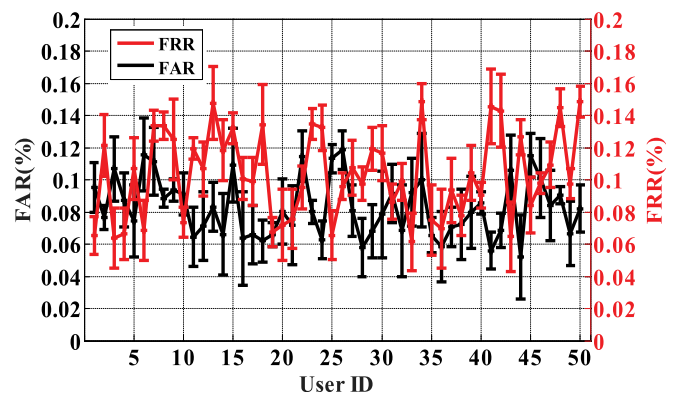


Fig. 8. FAR and FRR for 50 users.

TABLE 1
Breath Demarcation Accuracy with the Combination of Two Templates

UI	Age	Gender	Device	Motion Status	# of Breath found by ear	# of Correct Detection	# of Missing Detection
1	10-30	Male	Samsung Galaxy Note 2	Running	16	16	0
2		Female	Huawei Honor 3C	Sitting	26	26	0
3	30-40	Male	MIUI 4	Running	21	21	0
4		Female	Meizu 4	Sitting	18	17	1
5	40-60	Male	Huawei Honor 7	Sitting	14	14	0
6		Female	HTC One M8	Sitting	16	16	0

Breath Demarcation Accuracy of 50 users: 99.6%

Legend: UI = User ID

replay attack included one-time recording of 02 min duration through the user's smartphone. For time consistency, the 02 min recording was carried out once per week over a period of 08 months.

4.6 Evaluation Matrices

4.6.1 BreathID Evaluation

It involves recorded voices from 50 users, and aims for the following:

- *Breath Demarcation Accuracy*: We evaluate the accuracy of breath extraction algorithm to accurately and successfully extract the breath. It is cross-validated by visual inspection of audio spectrogram.
- *Speaker Identification*: This can be thought as a task of finding who is talking from a set of breaths of known users. It is a 1 to N match where the breath is compared against N templates. The merit we choose for identification errors is FIR of a valid user.
- *Speaker Verification*: The verification problem is a 1 to 1 match where a speaker's breath is matched to a template (profile or model). We evaluate the verification accuracy in terms of FRR, FAR and ERR. The FRR is the probability that a true user is falsely identified as an impostor, FAR is the probability that an impostor is incorrectly recognized as a true user whereas EER is the value that FAR equals to FRR. Lower EER means higher authentication accuracy.
- *Required Breath Segments (recorded voice)*: To find out the minimum number of breath segments (and the required voice recording). We consider FAR and FRR rates for both Training and Testing phases.
- *System Overhead*: Computational time and real-time memory (RAM) required for BreathID. The storage space of breath template is also considered.

4.6.2 Practical Modalities

The evaluation of various practical scenarios includes the recording from 20 users, and considers the following:

- *Text-independence*: A one-time experiment in which users speak four different selected text contents.
- *Motion Status*: Users are asked to speak while sitting, and while having a walk.
- *Recording Equipment*: Recordings are performed through 03 smartphones, and 02 commercial microphones.

- *Recording Period*: Recordings are performed once per week, over a span of 08 months.
- *Multilingualism*: Users speak English and then Chinese Mandarin languages.
- *Re-Recording (Replay Attack)*: Recorded voice is transmitted through commercial speaker, and re-recorded through commercial microphone over a distance of 6 inches. The results are compared with EER values for the case once the genuine voice is recorded.

5 EVALUATION RESULTS

5.1 BreathID Performance

5.1.1 Breath Demarcation Accuracy

To make a generic breath template (Section 4.1.1), we randomly select voice recordings of one male and one female to extract 50 breath segments. The results are presented in Table 1 for 06 speakers for illustration. The demarcation accuracy (the number of correct detections divided by the total number of breath events) of the algorithm is 99.6 percent, and the missing detection rate (the number of missing detected breath events divided by the total number of breath events) is 0.4 percent. We highlight that once we only use breath segments of two male candidates, the demarcation accuracy was little lower, i.e., 97.3 percent. The results also show that our breath demarcation process is independent of the age, gender, recording device and motion status.

5.1.2 Speaker Identification

In Speaker Identification, we make a comparison among all 50 users in pairwise manner, as shown in Table 2. The first row and column show the ID of each user, and a value in the table means the similarity score between two users. The last column shows the FIR. In real experiments, we find that there is only 01 out of 50 users who is falsely identified, i.e., average FIR is 0.04 percent. From these results, we also observe that the range of similarity scores (in terms of MFCC) lies between 0.3521 to 0.4464. This means that the lowest similarity (i.e., 0.3521) can be set as the threshold for Speaker Recognition phase.

5.1.3 Speaker Verification

The FAR and FRR results for Speaker Verification are shown in Figs. 7c and 7d in terms of time of speech required to achieve decimal level false rates, along with a comparison of Decision Maker with GMM, HMM, SVM, ANN and KNN

TABLE 2
User Identification Accuracy in MFCC

S \ UI	1	2	3	4	5	6	7	8	...	43	44	45	46	47	48	49	50	FIR (%)
1	0.38	0.12	0.15	0.07	0.12	0.09	0.05	0.10	...	0.08	0.02	0.08	0.18	0.14	0.11	0.16	0.12	0
2	0.21	0.41	0.21	0.14	0.11	0.11	0.10	0.15	...	0.04	0.27	0.17	0.31	0.25	0.17	0.24	0.19	0
3	0.26	0.21	0.44	0.15	0.14	0.19	0.13	0.17	...	0.01	0.13	0.10	0.26	0.35	0.22	0.12	0.23	0
4	0.26	0.29	0.25	0.42	0.26	0.29	0.23	0.29	...	0.15	0.22	0.33	0.26	0.17	0.16	0.21	0.32	0
5	0.06	0.04	0.07	0.09	0.35	0.12	0.04	0.05	...	0.01	0.07	0.11	0.02	0.01	0.01	0.02	0.05	0
6	0.12	0.10	0.16	0.13	0.14	0.36	0.13	0.12	...	0.08	0.26	0.25	0.06	0.04	0.04	0.07	0.21	0
7	0.18	0.17	0.29	0.21	0.15	0.23	0.41	0.22	...	0.12	0.12	0.02	0.22	0.12	0.10	0.11	0.42	2
8	0.25	0.35	0.31	0.24	0.17	0.21	0.20	0.43	...	0.11	0.10	0.05	0.32	0.28	0.19	0.26	0.21	0
...
43	0.13	0.18	0.09	0.21	0.08	0.25	0.03	0.06	...	0.41	0.25	0.23	0.02	0.29	0.13	0.23	0.15	0
44	0.09	0.21	0.10	0.22	0.06	0.24	0.05	0.21	...	0.31	0.39	0.16	0.14	0.13	0.14	0.21	0.22	0
45	0.17	0.18	0.05	0.21	0.08	0.24	0.13	0.12	...	0.06	0.04	0.41	0.04	0.09	0.11	0.08	0.04	0
46	0.09	0.18	0.14	0.03	0.02	0.01	0.02	0.03	...	0.03	0.21	0.06	0.40	0.34	0.24	0.18	0.14	0
47	0.03	0.07	0.08	0.00	0.01	0.01	0.01	0.01	...	0.07	0.09	0.25	0.12	0.45	0.20	0.10	0.05	0
48	0.07	0.10	0.11	0.00	0.01	0.00	0.00	0.01	...	0.03	0.14	0.17	0.20	0.41	0.42	0.15	0.09	0
49	0.16	0.21	0.14	0.04	0.03	0.04	0.03	0.06	...	0.18	0.15	0.23	0.25	0.34	0.26	0.44	0.18	0
50	0.05	0.07	0.09	0.02	0.02	0.03	0.04	0.03	...	0.06	0.28	0.18	0.07	0.07	0.04	0.03	0.42	0

Average Accuracy of 50 users: 99.96%, False Identification Rate of 50 users: 0.04%

Legend: S = Similarity; UI = User ID

algorithms. Next, Fig. 8 shows that both FAR and FRR results have negligible tolerances within 24 recorded voices for all 50 users. Lastly, Fig. 9b shows the verification accuracy (relationship between FRR and FAR) in terms of the minimum number of breath segments (and corresponding recorded speech) for 50 users. We observe that as the number of breath segments (length of recorded voice) increases, the false rates reduce in a drastic manner.

5.1.4 Required Breath (and Voice) Samples

We give answer to one of the critical questions: How much long recording (and breath samples) do we need to achieve a certain level of accuracy (FAR and FRR), both for the construction of Speaker’s Profile (Training Phase) and Speaker Verification (Testing Phase). Fig. 9 shows the results in a comprehensive way. We consider 05 to 90 sec of recorded voice, for which the extracted breath segments are shown in vertical axis for all 50 users. The graph on right shows the corresponding FAR and FRR results. Based on the results,

BreathID can achieve 0.08 percent FAR and 0.09 percent FRR for 60 sec of recorded voice (15 breath segments on average) during training phase. The false rates reduce to marginal values if the 90 sec speech is processed (22 breath segments). Similar results are observed during testing phase. A 20 sec recording results to 05 breath samples on average, which gives 0.12 percent FAR and 0.15 percent FRR. Both false rates almost reduce to zero if recorded speech over 90 sec (22 breaths) is considered.

5.1.5 System Overhead

The main overhead of our system lies in computational time and storage space. The time delay includes the overhead for breath demarcation, feature extraction and recognition phases. The breath demarcation overhead further includes the time for user profiling and recognition phases, which are primarily determined by the length of recorded speech in both phases. Therefore, we show the time and storage overhead in accordance with different lengths of input

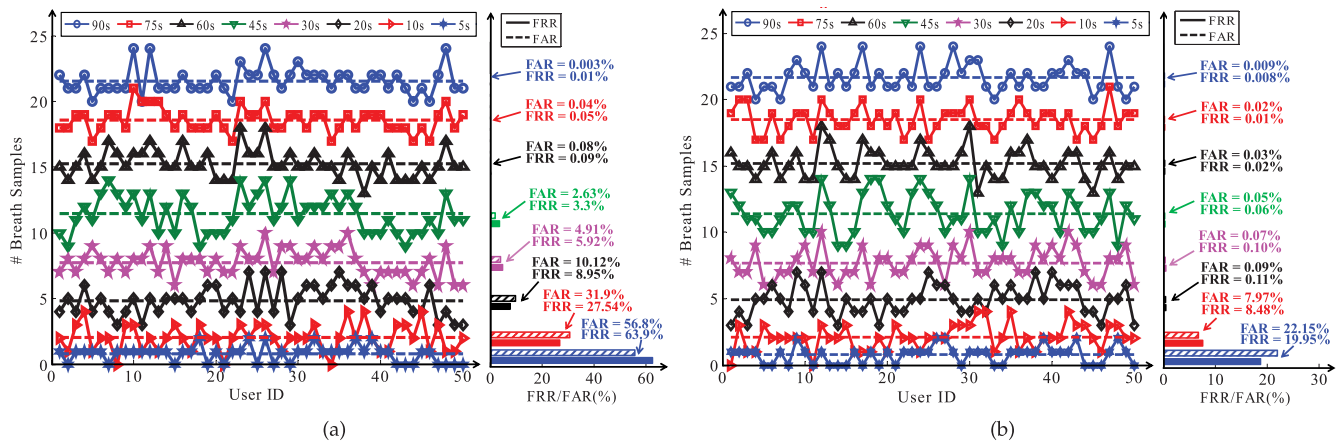


Fig. 9. No. of breath segments, voice length and false rates for training and testing phases. (a) Training phase (profiling); (b) testing phase (verification).

TABLE 3
System Overhead

Smartphone: SamSung Galaxy Note2, Android 4.3, Audio Format: 16-bit PCM 44 kHz Mono 705 Kbps WAV
Laptop: Corei3 (1.7 GHz, 2 GB RAM, 3 MB Cache), Matlab 7.6 (R2008a)

Time period (s) of speech		Time (ms) of Breath Demarcation		Time (ms) of Feature Extracting		Time (ms) of Classification	Time (ms) of Recognition	Storage (Kb) of Breath Profile
P	V	P	R	P	R			
30	10	362	114	68	23	14	177	180
60	20	651	207	124	39	14	260	423
120	30	1147	356	236	71	15	442	734

Legend: P = Profiling; V = Verification; R = Recognition

speech. The results are shown in Table 3. We can see that our system is time efficient which can verify a user within 260 msec if we use 60 sec of speech during Training Phase (profiling), and 20 sec speech for Testing Phase (recognition). Our system is quite handy as a single user profile only requires an average storage space of 423 Kbytes.

5.2 Considering Diverse Practical Modalities

As overall, our system shows promising results for various practical scenarios. During tests for *text-independence*, the results in Figs. 10d and 10e show that breath biometrics are hardly effected by the spoken text. The underlying reason is that the breath is an anatomical fingerprint of respiratory system, as discussed in detail in Section 2. For *Motion Scenario*, we consider two basic scenarios: the user is sitting, and having a brick walk, for which the results are shown in Figs. 10d and 10e. The effect of other scenarios on BreathID like exercise, running, respiratory disorder etc, can be addressed in future extension to our work. Next, Figs. 10d and 10e show the robustness of BreathID to degradation introduced by *recording equipment*. For this experiment, we considered 03 smartphone-embedded microphones (Samsung Galaxy Note2, Huawei Honor 3C, HTC ONE M8) and

02 stand-alone microphones. The better results by latter are due to high sensitivity, wide spectral coverage and low noise features of the microphones, the discussion of which is beyond the scope of this research.

To validate the consistency of breath biometric over significant time, we evaluate our scheme on 20 users over a period of 08 months. The voices are recorded each week. We achieve appealing results, as shown in Fig. 10a. The results for first and last month (July and February, respectively) are shown with vertical bars, while the min/max tolerance is shown for interim 06 months. We however mention that breath biometrics may change over large time, like a decade or two, as the body anatomy changes. For *Multilingualism*, we consider English and Chinese Mandarin languages. The volunteers speak random sentences in both languages to evaluate the system. Because of the anatomical nature of breath signals, only the negligible variation exists between EER values of all the users (Fig. 10b).

Lastly, an adversary can record the voice of a valid user to launch a *replay attack*. We replicate this scenario by transmitting the recorded voice of 20 registered users through a commercial speaker, and re-record it through a commercial microphone over a distance of 6 in. Fig. 10c compares the

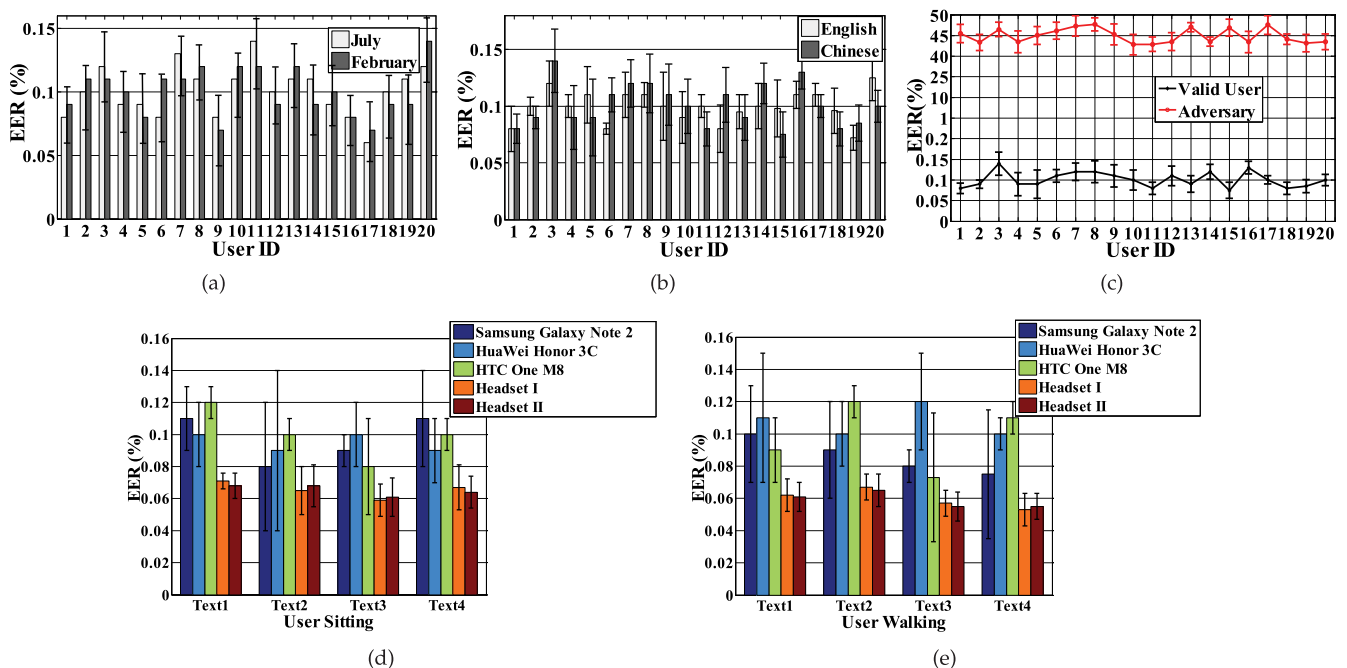


Fig. 10. The EER results of 20 users under various practical modalities. (a) Recording period; (b) multilingualism; (c) replay attack; (d) text-independence, sitting, microphone; (e) text-independence, walking, microphone.

EER of re-recorded speech with the results once the same user records his voice. The results show an average EER of 44.75 percent for 20 users, while the average EER with their original voice is 0.15 percent. As discussed in Section 2, the low-power, low-frequency and transient breath signals are highly degraded by the speaker-air-microphone channel, which leads to low recognition accuracy.

6 CONCLUSION AND FUTURE WORK

This paper introduces BreathID—a Speaker Recognition system based upon breath biometrics. We provide a detailed discussion for the design of our system under three parts: breath demarcation, feature extraction and decision maker. We show that unique breath features can be formulated for a Speaker Recognition system with promising Identification and Verification results. Though a preliminary work, we perform comprehensive evaluations on two aspects. We first evaluate BreathID for various parameters. We next examine our system for various modalities to show its practicability under real-life scenarios.

We foresee a number of future explorations to our work. These include the use of different MFCC hybrids for more refined feature extraction; fusion of other features with MFCC; optimization of BreathID with pattern matching and machine learning algorithms; combination of breath biometrics with exiting speech-based schemes for a two-fold, highly accurate and robust system. The noise analysis with varying degree of SNR and various noise sources will also be carried out over large datasets. For denoising or suppressing its effect, we plan to investigate three types of noise compensation techniques including filtering, noise model and empirical based techniques.

ACKNOWLEDGMENTS

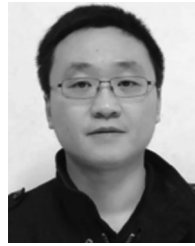
This work is supported by the National Key R&D Program of China (2017YFB1003003), and National Natural Science Foundation of China (61472068).

REFERENCES

- [1] I. McGraw, et al., "Personalized speech recognition on mobile devices," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 5955–5959.
- [2] K. Tokuda and H. Zen, "Directly modeling voiced and unvoiced components in speech waveforms by neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 5640–5644.
- [3] P. Buchner, S. Goronzy, R. Kompe, and S. Rapp, "Speech recognition control of remotely controllable devices in a home network environment," US Patent 6,535,854, Mar. 18 2003.
- [4] G. Heigold, I. Moreno, S. Bengio, and N. M. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 5115–5119.
- [5] R. Prabhavalkar, O. Alsharif, A. Bruguier, and I. McGraw, "On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, 2016, pp. 5970–5974.
- [6] I. Kelly, F. Boland, and J. Skoglund, "Robust estimation of reverberation time using polynomial roots," in *Proc. AES 60th Conf. Dereverberation Reverberation Audio Music Speech*, 2016, pp. 1–7.
- [7] S. G. Koolagudi and K. S. Rao, "Emotion recognition from speech using source, system, and prosodic features," *Int. J. Speech Technol.*, vol. 15, no. 2, pp. 265–289, 2012.
- [8] T. Arai, "History of chiba and kajiyama and their influence in modern speech science," *Proc. From Sound Sense*, vol. 50, pp. 115–120, 2004.
- [9] L. R. Rabiner and R. W. Schafer, "Introduction to digital speech processing," *Foundations Trends Signal Process.*, vol. 1, no. 1, pp. 1–194, 2007.
- [10] H. Aghajan, J. C. Augusto, and R. L.-C. Delgado. *Human-centric Interfaces for Ambient Intelligence*. San Diego, CA, USA: Academic Press, 2009.
- [11] R. D. Kent, C. Read, and R. D. Kent, *The Acoustic Analysis of speech*, vol. 58. San Diego, CA, USA: Singular Publishing Group, 1992.
- [12] A. Spanias, T. Painter, and V. Atti, *Audio Signal Processing and Coding*. Hoboken, NJ, USA: Wiley, 2006.
- [13] D. B. Roe, et al., *Voice Communication Between Humans and Machines*. Washington, DC, USA: National Academies Press, 1994.
- [14] P. Esquef, M. Karjalainen, and V. Välimäki, "Detection of clicks in audio signals using warped linear prediction," in *Proc. 14th Int. Conf. Digital Signal Process.*, vol. 2, 2002, pp. 1085–1088.
- [15] D. Ruinskiy and Y. Lavner, "An effective algorithm for automatic detection and exact demarcation of breath sounds in speech and song signals," *IEEE Int. Audio Speech Language Process.*, vol. 15, no. 3, pp. 838–850, Mar. 2007.
- [16] L. S. Kennedy and D. P. W. Ellis, "Pitch-based emphasis detection for characterization of meeting recordings," in *Proc. IEEE Workshop Autom. Speech Recog. Understanding.*, 2003, pp. 243–248.
- [17] M. S. Spina and V. W. Zue, "Automatic transcription of general audio data: Preliminary analyses," in *Proc. 4th Int. Conf. Spoken Language*, vol. 2, 1996, pp. 594–597.
- [18] C. W. Wightman and J. Bachenko, "Speech-recognition-assisted selective suppression of silent and filled speech pauses during playback of an audio recording," U.S. Patent 6161087, Dec. 12, 2000.
- [19] C. Paccalin and M. Jeannerod, "Changes in breathing during observation of effortful actions," *Brain Res.*, vol. 862, no. 1, pp. 194–200, 2000.
- [20] A. Rochet-Capellan and S. Fuchs, "Changes in breathing while listening to read speech: The effect of reader and speech mode," *Frontiers Psychology*, vol. 4, 2013, Art. no. 906.
- [21] F. Bergamin, "A fingerprint of exhaled breath," presented at the *ETH Life, News Rep.*, Zurich, Switzerland, Apr. 04, 2013.
- [22] V. B. Zordan, B. Celly, B. Chiu, and P. C. DiLorenzo, "Breathe easy: Model and control of human respiration for computer animation," *Graphical Models*, vol. 68, no. 2, pp. 113–132, 2006.
- [23] V. I. Korenbaum, "Features of acoustic processes in human respiratory system," Technical Note, Institute of Physics and Information Technologies, Far Eastern State University, Vladivostok, Russia, 1999, <http://www.akin.ru/Docs/Rao/Ses10/Me1.PDF>, Accessed: Nov. 2017.
- [24] T. Nakano, J. Ogata, M. Goto, and Y. Hiraga, "Analysis and automatic detection of breath sounds in unaccompanied singing voice," in *Proc. Int. Conf. Materials Process. Characterization*, vol. 10, 2008, pp. 387–390.
- [25] T. Chiba and M. Kajiyama, *The Vowel: Its Nature and Structure*. Tokyo, Japan: Tokyo-Kaiseikan, 1941.
- [26] K. Maekawa and K. Honda, "On the vowel, its nature and structure and related works by chiba and kajiyama," *J. Phonetic Soc. Japan*, vol. 5, no. 2, pp. 15–30, 2001.
- [27] K. N. Stevens. *Acoustic phonetics*, The MIT Press, 2000, ISBN: 0262692503.
- [28] V. Atti, "Algorithms and software for predictive and perceptual modeling of speech," *Synthesis Lectures Algorithms and Softw. Eng.*, vol. 2, no. 1, pp. 1–119, 2011.
- [29] B. Yegnanarayana and P. Satyanarayana Murthy, "Source-system windowing for speech analysis and synthesis," *IEEE Trans. Speech Audio Process.*, vol. 4, no. 2, pp. 133–137, Mar. 1996.
- [30] B. Wildermoth and K. K. Paliwal, "Use of voicing and pitch information for speaker recognition," in *Proc. 8th Australian Int. Conf. Speech Sci. Technol.*, 2000, pp. 324–328.
- [31] T. Starner and J. A. Paradiso, "Human generated power for mobile electronics, Low power electronics design, vol. 45, pp. 1–35, 2004.
- [32] A. Delnavaz and J. Voix, "Electromagnetic micro-power generator for energy harvesting from breathing," in *Proc. 38th Annu. Conf. IEEE Ind. Electronics Soc.*, 2012, pp. 984–988.
- [33] M. Sky, *Breathing: Expanding Your Power and Energy*. Rochester, VT, USA: Inner Traditions/Bear & Co, 1990.
- [34] J. H. Ryalls and S. Behrens, *Introduction to Speech Science: From Basic Theories to Clinical Applications*. Boston, MA, USA: Allyn & Bacon, 2000.

- [35] C. Maier, V. Rödler, H. Wenz, and H. Dickhaus, "ECG fingerprints of obstructed breathing in sleep apnea patients," *IEEE Eng. Medicine Biology Mag.: Quarterly Magazine Eng. Med. Biol. Soc.*, vol. 28, no. 6, pp. 41–48, Nov./Dec. 2008.
- [36] M. Green, J. Mead, and T. A. Sears, "Muscle activity during chest wall restriction and positive pressure breathing in man," *Respiration Physiology*, vol. 35, no. 3, pp. 283–300, 1978.
- [37] J. H. Green and J. B. L. Howell, "The correlation of intercostal muscle activity with respiratory air flow in conscious human subjects," *J. Physiology*, vol. 149, no. 3, 1959, Art. no. 471.
- [38] S. Jung, R. Thewes, T. Scheiter, K. F. Goser, and W. Weber, "A low-power and high-performance CMOS fingerprint sensing and encoding architecture," *IEEE J. Solid-State Circuits*, vol. 34, no. 7, pp. 978–984, Jul. 1999.
- [39] C. D. Patel, S. Trivedi, and S. Patel, "Biometrics in IRIS technology: A survey," *Int. J. Scientific Res. Publications*, vol. 2, pp. 3–4, 2012.
- [40] A. Hadid, J. Y. Heikkilä, O. Silvén, and M. Pietikainen, "Face and eye detection for person authentication in mobile phones," in *Proc. 1st ACM/IEEE Int. Conf. Distrib. Smart Cameras*, 2007, pp. 101–108.
- [41] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation Comput. Syst.*, vol. 16, no. 4, pp. 351–359, 2000.
- [42] F. Bergadano, D. Gunetti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Trans. Inf. Syst. Security*, vol. 5, no. 4, pp. 367–397, 2002.
- [43] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," in *Proc. 18th ACM Conf. Comput. Commun. Security*, 2011, pp. 139–150.
- [44] Y. Nakkabi, I. Traoré, and A. A. E. Ahmed, "Improving mouse dynamics biometric performance using variance reduction via extractors with separate features," *IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans*, vol. 40, no. 6, pp. 1345–1353, Nov. 2010.
- [45] M. Shahzad, A. X. Liu, and A. Samuel, "Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 39–50.
- [46] N. Zheng, K. Bai, H. Huang, and H. Wang, "You are how you touch: User verification on smartphones via tapping behaviors," in *Proc. IEEE 22nd Int. Conf. Netw. Protocols*, 2014, pp. 221–232.
- [47] C. Bo, L. Zhang, and X.-Y. Li, "Silentsense: Silent user identification via touch and movement behavioral biometrics," in *Proc. 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 187–190.
- [48] P. Saravanan, S. Clarke, D. H. Chau, and H. Zha, "Latentgesture: Active user authentication through background touch analysis," in *Proc. 2nd Int. Symp. Chin.*, 2014, pp. 110–113.
- [49] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smart phones," in *Proc. 20th Netw. Distrib. Syst. Security Symp. NDSS'13*, Internet Society, Reston, VA, 2013.
- [50] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi, "Tips: Context-aware implicit user identification using touch screen in uncontrolled environments," in *Proc. 15th Workshop Mobile Comput. Syst. Appl.*, 2014, Art. no. 9.
- [51] S. Nandyal, S. S. Wali, and S. M. Hatture, "MFCC based text-dependent speaker identification using BPNN," *Int. J. Signal Proc. Syst.*, vol. 3, no. 1, pp. 30–34, 2015.
- [52] A. Roy, M. M. Doss, and S. Marcel, "A fast parts-based approach to speaker verification using boosted slice classifiers," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 1, pp. 241–254, Feb. 2012.
- [53] A. Kabir and S. M. Masudul Ahsan, "Vector quantization in text dependent automatic speaker recognition using mel-frequency cepstrum coefficient," in *Proc. 6th WSEAS Int. Conf. Circuits, Syst., Electronics Control Signal Process. Cairo Egypt*, 2007, pp. 352–355.
- [54] A. Larcher, K. Aik Lee, B. Ma, and H. Li, "Text-dependent speaker verification: Classifiers, databases and RSR2015," *Speech Commun.*, vol. 60, pp. 56–77, 2014.
- [55] K. Brunet, K. Taam, E. Cherrier, N. Faye, and C. Rosenberger, "Speaker recognition for mobile user authentication: An android solution," in *Proc. 8ème Conférence sur la Sécurité des Architectures Réseaux et Systèmes d'Inform.*, 2013, Art. no. 10.
- [56] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Commun.*, vol. 52, no. 1, pp. 12–40, 2010.
- [57] A. Maesa, F. Garzia, M. Scarpiniti, and R. Cusani, "Text independent automatic speaker recognition system using mel-frequency cepstrum coefficient and gaussian mixture models," *J. Inf. Security*, vol. 3, no. 04, 2012, Art. no. 335.
- [58] T. Barbu, "A supervised text-independent speaker recognition approach," *Vectors*, vol. 2, 2007, Art. no. 6.
- [59] D. A. Reynolds, "Experimental evaluation of features for robust speaker identification," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 4, pp. 639–643, 1994.
- [60] Z. Qian, L.-Y. Liu, and X.-Y. Li, "Speaker identification based on MFCC and IMFCC," in *Proc. 1st Int. Conf. Inf. Sci. Eng.*, 2009, pp. 5416–5419.
- [61] S. Chatterjee, C. Koniaris, and W. B. Kleijn, "Auditory model based optimization of MFCCS improves automatic speech recognition performance," in *Proc. 10th Annu. Conf. Int. Speech Commun. Assoc.*, 2009, pp. 2943–2946.
- [62] K. A. Senthildevi and E. Chandra, "Keyword spotting system for tamil isolated words using multidimensional MFCC and DTW algorithm," in *Proc. Int. Conf. Commun. Signal Process.*, 2015, pp. 0550–0554.
- [63] D. Jianli and Y. Yong, "Noise detection algorithm based on modified-MFCC method," *J. Convergence Inf. Technol.*, vol. 7, no. 19, pp. 390–397, 2012.
- [64] K. Matsumoto, N. Hayasaka, and Y. Iiguni, "Noise robust speaker identification by dividing MFCC," in *Proc. 6th Int. Symp. Commun. Control Signal Process.*, 2014, pp. 652–655.
- [65] W. Hong and P. Jin Gui, "Modified MFCCs for robust speaker recognition," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, 2010, pp. 276–279.
- [66] B. Ayoub, K. Jamal, and Z. Arsalane, "An analysis and comparative evaluation of MFCC variants for speaker identification over VoIP networks," in *Proc. World Congr. Inf. Technol. Comput. Appl. Congr.*, 2015, pp. 1–6.
- [67] B. G. Nagaraja and H. S. Jayanna, "Efficient window for monolingual and crosslingual speaker identification using MFCC," in *Proc. Int. Conf. Adv. Comput. Commun. Syst.*, 2013, pp. 1–4.
- [68] A. Ghosal, R. Chakraborty, B. C. Dhara, and S. K. Saha, "Music classification based on MFCC variants and amplitude variation pattern: A hierarchical approach," *Int. J. Signal Process. Image Process. Pattern Recognit.*, vol. 5 no. 1, pp. 131–150, 2012.
- [69] S. Nakagawa, L. Wang, and S. Ohtsuka, "Speaker identification and verification by combining MFCC and phase information," *IEEE Trans. Audio Speech Language Process.*, vol. 20, no. 4, pp. 1085–1095, May 2012.
- [70] A. Bakshi, S. K. Koppurapu, S. Pawar, and S. Nema, "Novel windowing technique of MFCC for speaker identification with modified polynomial classifiers," in *Proc. 5th Int. Conf. Confluence Next Generation Inf. Technol. Summit*, 2014, pp. 292–297.
- [71] U. Bhattacharjee and K. Sarmah, "Language identification system using MFCC and prosodic features," in *Proc. Int. Conf. Intell. Syst. Signal Process.*, 2013, pp. 194–197.
- [72] S. Chakraborty, A. Roy, and G. Saha, "Fusion of a complementary feature set with MFCC for improved closed set text-independent speaker identification," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2006, pp. 387–390.
- [73] P. M. Chauhan and N. P. Desai, "Mel frequency cepstral coefficients (MFCC) based speaker identification in noisy environment using wiener filter," in *Proc. Int. Conf. Green Comput. Commun. Elect. Eng.*, 2014, pp. 1–5.
- [74] S. Pandiaraj, H. N. R. Keziah, D. S. Vinothini, L. Gloria, and K. R. Shankar Kumar, "A confidence measure based—Score fusion technique to integrate MFCC and pitch for speaker verification," in *Proc. 3rd Int. Conf. Electron. Comput. Technol.*, 2011, pp. 317–320.
- [75] R. Ajgou, S. Sbaa, S. Ghendir, A. Chamsa, and A. Taleb-Ahmed, "Robust remote speaker recognition system based on AR-MFCC features and efficient speech activity detection algorithm," in *Proc. 11th Int. Symp. Wireless Commun. Syst.*, 2014, pp. 722–727.
- [76] H. Trang, T. H. Loc, and H. B. H. Nam, "Proposed combination of PCA and MFCC feature extraction in speech recognition system," in *Proc. Int. Conf. Adv. Technol. Commun.*, 2014, pp. 697–702.
- [77] Z. Wu and Z. Cao, "Improved MFCC-based feature for robust speaker identification," *Tsinghua Sci. Technol.*, vol. 10, no. 2, pp. 158–161, 2005.
- [78] L. Oudre, "Automatic detection and removal of impulsive noise in audio signals," *Image Process. On Line*, vol. 5, pp. 267–281, 2015.
- [79] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*, 1st Ed., Pearson Education Inc. 1993.
- [80] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 28, no. 4, pp. 357–366, Aug. 1980.

- [81] L. S. Kennedy and D. P. W. Ellis, "Laughter detection in meetings," in *Proc. NIST Int. Conf. Acoust. Speech Signal Process.*, 2004, pp. 118–121.
- [82] R. Cai, L. Lu, H.-J. Zhang, and L.-H. Cai, "Highlight sound effects detection in audio stream," in *Proc. Int. Conf. Multimedia Expo*, 2003, pp. 37–40.
- [83] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ, USA: Prentice Hall, 1978.
- [84] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2012.
- [85] A. V. Oppenheim and R. W. Schafer, "From frequency to quefrency: A history of the cepstrum," *IEEE Signal Process. Mag.*, vol. 21, no. 5, pp. 95–106, Sep. 2004.
- [86] S. S. Stevens, J. Volkman, and E. B. Newman, "The mel scale equates the magnitude of perceived differences in pitch at different frequencies," *J. Acoust. Soc. America*, vol. 8, no. 3, pp. 185–190, 1937.
- [87] S. Lindholm, "A speech recognition system for swedish running on Android," MSc. Thesis, Department of Computer Science, Lund university, LTH, 2010.
- [88] S. Sigurdsson, K. B. Petersen, and T. Lehn-Schiøler, "Mel frequency cepstral coefficients: An evaluation of robustness of MP3 encoded music," in *Proc. 7th Int. Conf. Music Inf. Retrieval*, 2006, pp. 21–26.
- [89] K. K. Bhuvanagiri and S. K. Kopparapu, "Modified mel filter bank to compute MFCC of subsampled speech," *arXiv preprint arXiv:1410.7382*, 2014.
- [90] K. K. Bhuvanagiri and S. K. Kopparapu, "Recognition of subsampled speech using a modified mel filter bank," in *Proc. Int. Conf. Advances Comput. Commun.*, 2011, pp. 293–299.
- [91] M. Jalil, F. A. Butt, and A. Malik, "Short-time energy, magnitude, zero crossing rate and autocorrelation measurement for discriminating voiced and unvoiced segments of speech signals," in *Proc. Int. Conf. Technological Advances Elect., Electron. Comput. Eng.*, 2013, pp. 208–212.
- [92] K. Iwano, T. Asami, and S. Furui, "Noise-robust speaker verification using f0 features," in *Proc. INTERSPEECH- Int. Conf. Spoken Language Process.*, 2004, vol. 2, pp. 1417–1420.
- [93] G. Fant, *Acoustic Theory of Speech Production: With Calculations Based on X-Ray Studies of Russian Articulations*, Number 2. Berlin, Germany: Walter de Gruyter, 1970.
- [94] D. H. Klatt, "Software for a cascade/parallel formant synthesizer," *J. Acoustical Soc. America*, vol. 67 no. 3, pp. 971–995, 1980.
- [95] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Dig. Signal Process.*, vol. 10, no. 1, pp. 19–41, 2000.
- [96] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Process.*, vol. 10, no. 1, pp. 42–54, 2000.
- [97] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Comput. Speech Language*, vol. 20, no. 2, pp. 210–229, 2006.
- [98] T. Matsui and S. Furui, "Comparison of text-independent speaker recognition methods using vq-distortion and discrete/continuous hmm's," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 3, pp. 456–459, Jul. 1994.
- [99] K. R. Farrell, R. J. Mammone, and K. T. Assaleh, "Speaker recognition using neural networks and conventional classifiers," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 1, pp. 194–205, Jan. 1994.
- [100] P. Motlicek, S. Dey, S. Madikeri, and L. Burget, "Employment of subspace gaussian mixture models in speaker recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 4445–4449.
- [101] J. Patel and A. Nandurbarkar, "Development and implementation of algorithm for speaker recognition for gujarati language," *Int. Res. J. Eng. Technol.*, vol. 2, no. 2, pp. 444–448, 2015.
- [102] J. Kamarauskas, "Speaker recognition using gaussian mixture models," *Elektronika ir Elektrotechnika*, vol. 85, no. 5, pp. 29–32, 2015.
- [103] B. Gold, N. Morgan, and D. Ellis, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Hoboken, NJ, USA: Wiley, 2011.



Li Lu (S'05-M'07) received the bachelor's and master's degrees from Zhejiang University, in 2000 and 2003, respectively, both in automation control. He received the PhD degree in the Key Lab of Information Security, Chinese Academy of Science, in 2007. He is a professor in the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC). His research interests include RFID technology and system, Wireless Security and Mobile Applications. He is a member of the IEEE Communication Society, the IEEE, and the ACM.



Lingshuang Liu received the bachelor's and master's degrees from the University of Electronic Science and Technology of China, in 2014 and 2017, respectively, both in computer science. Her research interests include mobile applications and wireless network security.



Muhammad Jawad Hussain received the BE degree in avionics from National University of Science and Technology (NUST), Pakistan, in 2005. He undertook the MS degree under information security followed by PhD degree in embedded systems from UESTC, in 2017. His research interests include security in RF backscatter tokens, Computational RFIDs, and Structural Health Monitoring systems.



Yongshuai Liu received the bachelor and master's degrees from the University of Electronic Science and Technology of China, in 2013 and 2016, respectively, both in computer science. His research interests include mobile applications and wireless network security.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Multi-User Multi-Keyword Rank Search Over Encrypted Data in Arbitrary Language

Yang Yang¹, Member, IEEE, Ximeng Liu², Member, IEEE, and Robert H. Deng³, Fellow, IEEE

Abstract—Multi-keyword rank searchable encryption (MRSE) returns the top- k results in response to a data user's request of multi-keyword search over encrypted data, and hence provides an efficient way for preserving data privacy in cloud storage systems while without loss of data usability. Many existing MRSE systems are constructed based on an algorithm which we term as k -nearest neighbor for searchable encryption (KNN-SE). Unfortunately, KNN-SE has a number of shortcomings, which limit its practical applications. In this paper, we propose a new MRSE system which overcomes almost all the defects of the KNN-SE based MRSE systems. Specifically, our new system does not require a predefined keyword set and supports keywords in arbitrary languages, is a multi-user system which supports flexible search authorization and time-controlled revocation, and it achieves better data privacy protection since even the cloud server is not able to tell which documents are the top- k results returned to a data user. We also conduct extensive experiments to demonstrate the efficiency of the new system.

Index Terms—Searchable encryption, multiple keyword, rank, top- k , privacy-preserving

1 INTRODUCTION

CLOUD computing [1] provides virtually unlimited computational and storage resources and has attracted increasing number of individuals and corporations to migrate their data into the cloud. The data privacy concerns the cloud computing brings along with it [2], [3] inspire cloud users to encrypt their sensitive documents before they are outsourced to cloud. Encryption translates data into unreadable ciphertext and how to search over and share encrypted data have been a challenging research problem. Searchable encryption (SE) [4], [5], [6] has been proposed as an effective method to execute keyword search over encrypted data. To make an encrypted document searchable in SE, a data owner first extracts a set of keywords from the document and encrypts them into an encrypted index. Then, the data owner uploads both the encrypted index and the encrypted document to the cloud for storage. In the data query phase, a data user creates a keyword token and submits the token to the cloud. The cloud uses a matching algorithm to test the association between the search token

and encrypted indices. Then, the encrypted documents with matching keywords are returned to the data user.

Many efforts have been spent to study SE systems in diverse application scenarios, such as health care [7], [8], smart grid [9], internet of things [10]. Many of these SE systems only support single keyword search [11], [12] or simple conjunctive queries [13], [14], [15], [16], and are not able to rank searched documents according to some predefined relevance score.

To provide an improved search experience, the multi-keyword rank searchable encryption (MRSE) mechanism is proposed [28], [29], [30], [32], which allows the cloud to return the top- k results (with the k highest relevance scores) rather than all relevant documents. However, most of the existing MRSE systems are based on a special k -nearest neighbor (KNN) algorithm [28], which we refer to as k -nearest neighbor algorithm for searchable encryption (KNN-SE) during the rest of the paper. Unfortunately, as we show in the paper, KNN-SE and hence the existing MRSE systems based on it suffer from many shortcomings, which greatly limit their practical applications. It is necessary to design new MRSE systems to overcome these defects without loss of efficiency and security.

1.1 Related Work

The concept of secure search over encrypted data was first proposed by Song et al. [17] in 2000. Curmola et al. [18] proposed a searchable symmetric encryption (SSE) scheme to secure storage system. Cash et al. [19] suggested a scalable SSE scheme supporting boolean query. Liu et al. [20] utilized the SSE and attribute based encryption to securely share and search for real-time video data. Yang et al. [21] proposed a non-interactive order-preserving encryption scheme to search over encrypted database system, which

- Y. Yang and X. Liu are with College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian 350002, China, The School of Information Systems, Singapore Management University, Singapore 188065, Singapore, The University Key Laboratory of Information Security of Network Systems (Fuzhou University), Fuzhou, Fujian Province 350002, China, The Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou, Fujian 350002, China, and also with the Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou, Fujian 350002, China. E-mail: {yang.yang.research, snbnix}@gmail.com.
- R.H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065, Singapore
E-mail: robertdeng@smu.edu.sg.

Manuscript received 11 May 2017; revised 8 Aug. 2017; accepted 21 Dec. 2017. Date of publication 27 Dec. 2017; date of current version 18 Mar. 2020.
(Corresponding author: Ximeng Liu.)

Digital Object Identifier no. 10.1109/TDSC.2017.2787588

supports range search. Zuo et al. [22] designed a SSE scheme to realize boolean search in secure database, which has higher efficiency compared with Cash's scheme [19]. Sun et al. [23] suggested a multi-client SSE system with supports boolean queries, which prevents the pre-query interaction between data owners and users. Kermanshahi et al. [24] also proposed a multi-user SSE scheme, which is constructed based on the oblivious cross tags protocol. Boneh et al. [25] put forth a public key encryption with keyword search scheme. Liang et al. [26] utilized the proxy re-encryption mechanism to design a public key searchable encryption scheme, and then they constructed a regular language search scheme over encrypted cloud data [27].

In 2011, Cao et al. [28] proposed the first framework of single-user MRSE based on KNN-SE. KNN-SE is a symmetric encryption system which utilizes "inner product similarity" to quantify the similarity and sort the result. A secret key in KNN-SE consists of two $k \times k$ matrices M_1, M_2 and a vector $S \in \{0, 1\}^k$, where k is the number of predefined keywords in the system. For each file, the extracted keywords are mapped to a vector $I \in \{0, 1\}^k$ such that each bit indicates whether a predefined keyword is in the file. Then, the vector I is split into two vectors I', I'' controlled by vector S . Finally, I', I'' is multiplied by M_1^T and M_2^T , respectively, to generate an encrypted index. The trapdoor generation is similar to the encrypted index except that the split query vector is multiplied with M_1^{-1} and M_2^{-1} . In the search phase, dot product is utilized to calculate the relevance score. (A brief overview of KNN-SE is given in Supplemental Materials A, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TDSC.2017.2787588>. The reader may refer to [28], [38] for more details.)

Since the publication of [28], most of the follow-on MRSE systems are constructed based on the KNN-SE approach. Yu et al. [29] put forth a two round searchable encryption system to realize top- k multi-keyword search. They employ KNN-SE (which is referred to as the vector space model in [29]) and order preserving encryption techniques to improve the security.

Fu et al. [30] suggested a multi-keyword rank search system supporting synonym query based on KNN-SE. It enables synonym queries such that the possible synonym substitution is tolerated in the data retrieval process. In [30], TF-IDF (term frequency-inverse document frequency) is leveraged in keyword extraction procedure. A data owner has to build an index tree to accelerate the search algorithm, which consumes more storage space. Later on, they proposed a verifiable keyword based semantic search system over encrypted data [31], which supports the verifiability of search result. They designed a symbol-based index tree to store the "path" information. The search result is verified using this tree.

Sun et al. [32] also propose a verifiable searchable encryption system to support multi-keyword search and similarity-based ranking. They utilize tree-based index structure, multi-dimension algorithm and KNN-SE to improve the search efficiency. Li et al. [33] integrate the KNN-SE and blind storage methods to construct an MRSE with blind storage. Then, they [34], [35] utilize superincreasing sequence to construct a new MRSE system supporting boolean keyword

query, such as "AND", "OR" and "NO" operations. They also leverage classified sub-dictionaries method to achieve better efficiency. Xia et al. [36] design a special tree-based index structure and a "greedy depth-first search" algorithm to enhance the search efficiency. They also used KNN-SE algorithm to encrypt the index and query.

Chen et al. [37] construct a hierarchical clustering method to enable more search semantics. Based on the minimum relevance threshold, the hierarchical method clusters encrypted documents and partitions the resulting clusters into sub-clusters. In that way, they achieve better search speed. Fu et al. [38] use locality sensitive hash function, Bloom filter and KNN-SE to realize a multi-keyword fuzzy searchable encryption system.

Although a lot of the MRSE systems are built based on the KNN-SE algorithm, the algorithm actually has several obvious limitations. First, KNN-SE requires a predefined keyword set at the system setup phase and the entire system need to be rebuilt in order to add any new keywords into the system. Second, KNN-SE is a symmetric key encryption system and hence a data owner has to disclose his secret key to a data user in order to authorize the latter query and decryption privileges and the authorization cannot be revoked even if the authorized data user is found behaving maliciously. Third, it's impossible for KNN-SE based MRSE systems to support keyword search in arbitrary languages since the number of keywords and hence the dimensions of the matrices M_1 and M_2 are astronomical in order to supports all keywords in all languages. Fourth, the relevance scores of the documents are in plaintext and the cloud server learns the statistic information of data, such as which are the high relevant documents and the high-frequency returned files. These information leaks user's privacy.

This brief defects analysis of KNN-SE brings us a simple conclusion: existing MRSE systems based on KNN-SE are not suitable for practical applications.

1.2 Contributions

In this paper we propose a new MRSE system which overcomes all the limitations of the KNN-SE based MRSE systems while without loss of efficiency and security. In particular, our new MRSE enjoys the following desirable properties.

- *No need for pre-defined keywords.* The new system does not require a set of pre-define keywords during the setup phase and new keywords can be added any time during the system operation.
- *Support arbitrary language.* We use Unicode [48] to encode keywords in arbitrary languages and utilize an efficient way to transform them into encrypted ciphertext.
- *Flexible authorization and time-controlled revocation.* The system allows a data owner to authorize a data user the search and decryption privileges in a specified period of time. When the current time is out of the defined time period, the right is automatically revoked. Moreover, the system also provides the data owner an effective way to deprive the authorized privileges within a time period.
- *Simultaneously search on multi-owner's data.* A data user can simultaneously search on multiple data

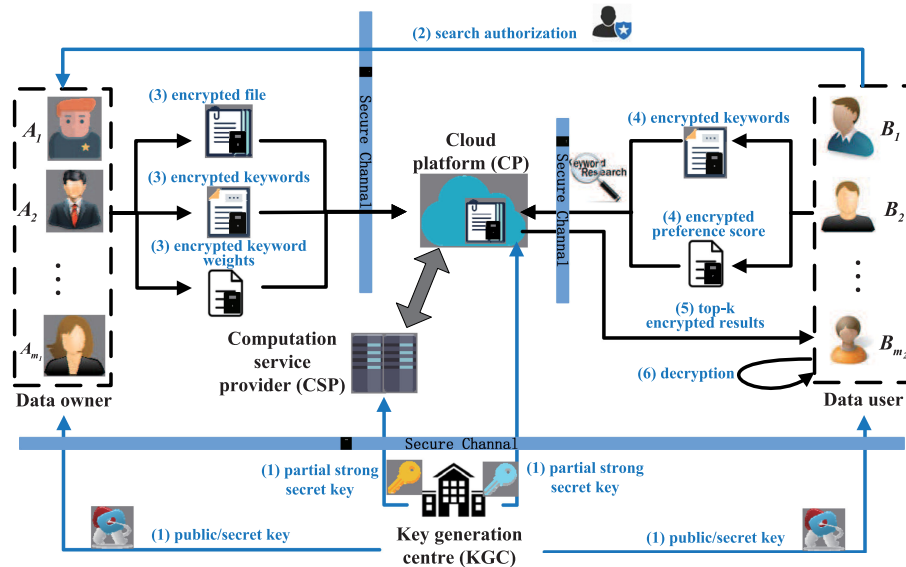


Fig. 1. System model.

owner's encrypted documents. For instance, a medical doctor can simultaneously search over encrypted medical records produced by different clinics. The other existing searchable encryption schemes have to generate different trapdoors to search over different data owner's documents. While, our scheme uses only one trapdoor to search on multiple owners' data.

- *Flexible keyword weight and preference setting.* In the encryption phase, a data owner sets different keyword weights according to the importance of these keywords. In the query phase, a data user searches on multiple keywords and sets different preference scores for each keyword. In the search phase, a cloud server computes the relevance scores in encrypted form according to the keyword weight and preference scores, and returns top- k results to the data user.
- *Security and Efficiency.* In the existing MRSE systems, the server is able to learn the plaintext of relevance scores of each searched document. A cloud server learns which documents are the most relevant. In our system, the cloud server learns nothing from the search results since the relevance scores returned to data user are encrypted. We prove the security of this system and conduct extensive simulations to demonstrate its efficiency.

2 SYSTEM ARCHITECTURE AND SECURITY MODEL

2.1 System Model

The system shown in Fig. 1 consists of five entities.

- *Key generation center (KGC)* is trusted by all entities in the system, who is responsible to generate keys for each entities in the system.
- *Cloud platform (CP)* has powerful storage and computation abilities and stores user's files in an encrypted form. CP also responds on users' computation and data retrieval requests.
- *Computing service provider (CSP)* is an online computation server and processes strong calculational capabilities. It executes interactive computations with CP.

- *Data owner.* Data owner encrypts his documents and outsources them to CP.
- *Data user.* Data user generates a keyword trapdoor to issue data retrieval request to CP.

2.2 Attack Model

We follow the attack model in [42], [43], in which KGC is a fully trusted entity, and CP and CSP are "honest-but-curious" who are honest to execute the protocols but curious with the plaintext of user data. An adversary \mathcal{A}^* is defined in this attack model, whose purpose is to recover the plaintext of data owner's privacy-preserving documents and data user's retrieved results. \mathcal{A}^* has the following abilities.

- 1) \mathcal{A}^* could *eavesdrop* all communications to get the transmitted information.
- 2) \mathcal{A}^* could *compromise* CP and try to get the plaintext from the encrypted information sent by the data owner and CSP.
- 3) \mathcal{A}^* could *compromise* CSP and try to obtain the plaintext from the ciphertext sent by CP in interactive protocol.
- 4) \mathcal{A}^* could *compromise* a set of data users (except the challenge user) and get their privileges. \mathcal{A}^* wants to get the plaintext information that belong to the challenge user.

However, the attacker \mathcal{A}^* is not allowed to compromise: (1) CP and CSP simultaneously, (2) the challenge user. These are typical restrictions in cryptographic protocols [44].

2.3 Security Model

The security model in [45], [46] is adopted in this work. Consider three parties: system user (a.k.a " D_1 "), CP (a.k.a " S_1 ") and CSP (a.k.a " S_2 "), where the system user includes data owner and user. We construct three simulators ($Sim_{D_1}, Sim_{S_1}, Sim_{S_2}$) against three types of attackers ($\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2}$) that corrupt D_1, S_1 and S_2 , respectively. These attackers are deemed as non-colluding and semi-honest. Due to the length limitation, please refer to [40], [45], [46] for the general security model definitions.

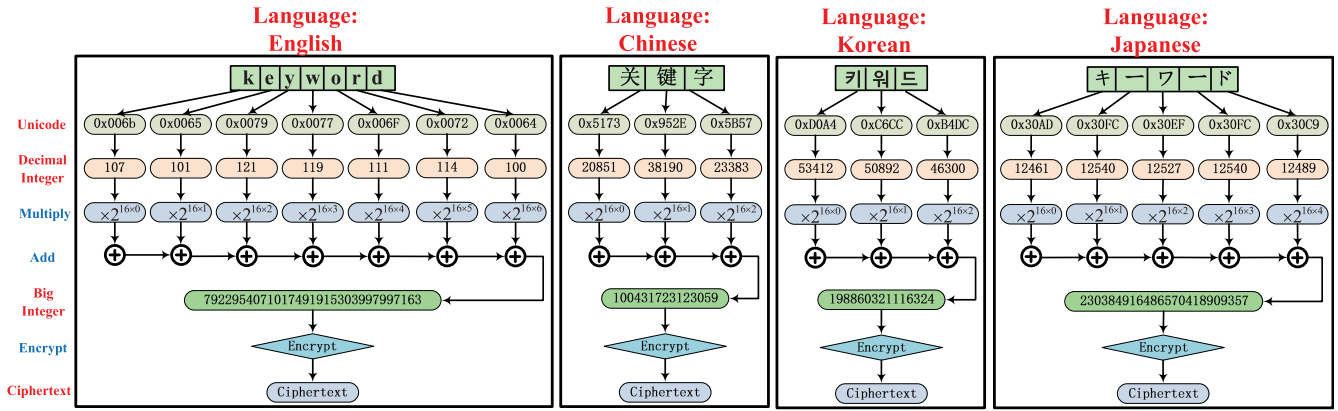


Fig. 2. Example of keyword to ciphertext.

3 CRYPTO PRIMITIVES AND PROTOCOLS

3.1 Paillier Cryptosystem with Threshold Decryption

The Paillier cryptosystem [39] with threshold decryption (PCTD) in [40], [41] is utilized for data encryption and could prevent the private key leakage risk in this paper. Let $\mathcal{L}(X)$ be the bit length of X .

KeyGen. Let k be the security parameter and p, q be two large prime numbers such that $\mathcal{L}(p) = \mathcal{L}(q) = k$. Let $N = pq$ and $\lambda = \text{lcm}(p-1, q-1)$.¹ Define a function $L(x) = \frac{x-1}{N}$ and select a generator g of order $\text{ord}(g) = (p-1)(q-1)/2$. The system public parameter is $PP = (g, N)$. The master secret key of the system is $SK = \lambda$. A user i in the system is assigned a secret key $sk_i \in Z_N$ and a public key $pk_i = g^{sk_i} \text{ mod } N^2$.

Encryption. On input a plaintext $m \in Z_N$, a user randomly selects $r \in [1, N/4]$ and uses his public key pk_i to encrypt m to ciphertext $[m]_{pk_i} = (C_1, C_2)$, in which $C_1 = pk_i^r(1 + mN) \text{ mod } N^2$ and $C_2 = g^r \text{ mod } N^2$.

Decryption with weak secret key. On input ciphertext $[m]_{pk_i}$ and weak private key sk_i , the message is recovered by computing $m = L(C_1/C_2^{sk_i} \text{ mod } N^2)$.

Decryption with master secret key. Using master secret key $SK = \lambda$ of the system, any ciphertext $[m]_{pk_i}$ encrypted by any public key can be decrypted by computing $C_1^\lambda = (pk_i^r)^\lambda(1 + mN\lambda) = (1 + mN\lambda) \text{ mod } N^2$. Since $\text{gcd}(\lambda, N) = 1$ holds,² we have $m = L(C_1^\lambda \text{ mod } N^2)\lambda^{-1} \text{ mod } N$.

Master secret key splitting. The master secret key $SK = \lambda$ is randomly split into two parts $SK_1 = \lambda_1$ and $SK_2 = \lambda_2$ such that $\lambda_1 + \lambda_2 \equiv 0 \text{ mod } \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \text{ mod } N^2$.

Partial decryption with SK_1 (PD1). On input the ciphertext $[m]_{pk_i} = (C_1, C_2)$, we use $SK_1 = \lambda_1$ to compute $C_1^{(1)} = (C_1)^{\lambda_1} = (pk_i^r)^{\lambda_1}(1 + mN\lambda_1) \text{ mod } N^2$.

Partial decryption with SK_2 (PD2). On input $[m]_{pk_i}$ and $C_1^{(1)}$, we use $SK_2 = \lambda_2$ to compute $C_1^{(2)} = (C_1)^{\lambda_2} = (pk_i^r)^{\lambda_2}(1 + mN\lambda_1) \text{ mod } N^2$. The message is recovered by computing $m = L(C_1^{(1)} \cdot C_1^{(2)})$.

Ciphertext refresh (CR): CR algorithm is utilized to refresh a ciphertext $[m]_{pk_i} = (C_1, C_2)$ into a new ciphertext $[m']_{pk_i} =$

1. *lcm* : lowest common multiple.
2. *gcd*: greatest common divider.

(C'_1, C'_2) such that $m = m'$. It selects a random $r' \in Z_N$, calculates $C'_1 = C_1 \cdot pk_i^{r'} \text{ mod } N^2$ and $C'_2 = C_2 \cdot g^{r'} \text{ mod } N^2$.

It is easy to verify that PCTD is *additive homomorphic*: $[m_1]_{pk_i} \cdot [m_2]_{pk_i} = [m_1 + m_2]_{pk_i}$, and *scalar-multiplicative homomorphic*: $([m]_{pk_i})^r = [r \cdot m]_{pk_i}$ for a random $r \in Z_N$.

The following two protocols in [40] are also utilized in the proposed system. Let pk_A and pk_B be the public keys of users A and B . pk_Σ is another public key that is defined later.

Secure addition protocol across domains (SAD): Given $[X]_{pk_A}$ and $[Y]_{pk_B}$, SAD protocol securely calculates $[X + Y]_{pk_\Sigma}$.

Secure multiplication protocol across domains (SMD): Given $[X]_{pk_A}$ and $[Y]_{pk_B}$, SMD protocol securely calculates $[X \cdot Y]_{pk_\Sigma}$.

3.2 Keyword Representation and Encryption

An important issue is how can we transform a keyword into a ciphertext to support any keyword in any language without predefined keyword set. We design a secure keyword to ciphertext algorithm (**K2C**) to fulfill the requirement, which mainly includes the following steps. (1) Map each character (including the special character) in the keyword into its unicode (UTF-16: 16-bit Unicode Transformation Format) [48]; (2) convert each hexadecimal unicode into decimal integer; (3) according to the position of the character in the keyword, multiply the decimal integer of the character with a weight; (4) add all the weighted decimal integer into a big integer; and (5) utilize the PCTD encryption algorithm and data owner's public key to encrypt the big integer of the keyword into a ciphertext.

An example is given in Fig. 2 to illustrate how to transform the string "keyword" in English, Chinese and Japanese languages into a ciphertext using K2C algorithm. It is worth noting that the K2C algorithm successfully converts a keyword into a unique big integer, which greatly solves the erroneous probability problem by using the bloom filter in other searchable encryption schemes [38], [47].

3.3 Encrypted Keyword Equivalence Testing Protocol

After keyword is encrypted, another crucial issue is to test whether two keyword ciphertexts contain the same keyword. A secure keyword equivalent test protocol across domains (**KET**) is designed to fulfill the task. Input two encrypted keywords $[X]_{pk_A}$ and $[Y]_{pk_B}$ that are encrypted by

different public keys, **KET** protocol outputs an encrypted result $[u^*]_{pk_\Sigma}$ to indicate whether these two keywords are the same. In order to make the protocol properly work, we require that $\mathcal{L}(X), \mathcal{L}(Y) < \mathcal{L}(N)/4$. CP and CSP jointly execute the protocol using the assigned keys SK_1 and SK_2 , respectively.

Step 1: CP computes

$$\begin{aligned} [X_1]_{pk_A} &= ([X]_{pk_A})^2 \cdot [1]_{pk_A} = [2X + 1]_{pk_A}; \\ [Y_1]_{pk_B} &= ([Y]_{pk_B})^2 = [2Y]_{pk_B}; \\ [X_2]_{pk_A} &= ([X]_{pk_A})^2 = [2X]_{pk_A}; \\ [Y_2]_{pk_B} &= ([Y]_{pk_B})^2 \cdot [1]_{pk_B} = [2Y + 1]_{pk_B}. \end{aligned}$$

Then, CP randomly selects r_1, r_2, r_3, r_4 such that $\mathcal{L}(r_1), \mathcal{L}(r_2) < \mathcal{L}(N)/4 - 1, \mathcal{L}(r_3), \mathcal{L}(r_4) < \mathcal{L}(N)/8$. Then, CP flips random coins $s_1, s_2 \in \{0, 1\}$.

CP and CSP jointly execute the following operations.

$$\text{If } s_1 = 1, [\gamma_1]_{pk_\Sigma} = \text{SAD}([X_1]_{pk_A}^{r_1}, ([Y_1]_{pk_B})^{N-r_1}).$$

$$\text{If } s_1 = 0, [\gamma_1]_{pk_\Sigma} = \text{SAD}([X_1]_{pk_A}^{N-r_1}, ([Y_1]_{pk_B})^{r_1}).$$

$$\text{If } s_2 = 1, [\gamma_2]_{pk_\Sigma} = \text{SAD}([X_2]_{pk_A}^{N-r_2}, ([Y_2]_{pk_B})^{r_2}).$$

$$\text{If } s_2 = 0, [\gamma_2]_{pk_\Sigma} = \text{SAD}([X_2]_{pk_A}^{r_2}, ([Y_2]_{pk_B})^{N-r_2}).$$

CP calculates $l_1 = [\gamma_1]_{pk_\Sigma} \cdot [r_3]_{pk_\Sigma}, l_2 = [\gamma_2]_{pk_\Sigma} \cdot [r_4]_{pk_\Sigma}, l'_1 = \text{PD1}_{SK_1}(l_1), l'_2 = \text{PD1}_{SK_1}(l_2)$ and sends (l_1, l'_1, l_2, l'_2) to CSP.

Step 2: CSP decrypts $l''_1 = \text{PD2}_{SK_2}(l_1, l'_1), l''_2 = \text{PD2}_{SK_2}(l_2, l'_2)$.

If $\mathcal{L}(l''_1) > \mathcal{L}(N)/2$, CSP sets $u'_1 = 0$ and $u'_1 = 1$ otherwise.

If $\mathcal{L}(l''_2) > \mathcal{L}(N)/2$, CSP sets $u'_2 = 0$ and $u'_2 = 1$ otherwise.

Then, CSP encrypts u'_1, u'_2 (with public key pk_Σ) to $([u'_1]_{pk_\Sigma}, [u'_2]_{pk_\Sigma})$, which are sent to CP.

Step 3: Receiving $([u'_1]_{pk_\Sigma}, [u'_2]_{pk_\Sigma})$, CP calculates as below.

If $s_1 = 1$, CP computes $[u_1]_{pk_\Sigma} = [u'_1]_{pk_\Sigma}$; otherwise, CP computes $[u_1]_{pk_\Sigma} = [1]_{pk_\Sigma} \cdot ([u'_1]_{pk_\Sigma})^{N-1} = [1 - u'_1]_{pk_\Sigma}$.

If $s_2 = 1$, CP computes $[u_2]_{pk_\Sigma} = [u'_2]_{pk_\Sigma}$; otherwise, CP computes $[u_2]_{pk_\Sigma} = [1]_{pk_\Sigma} \cdot ([u'_2]_{pk_\Sigma})^{N-1} = [1 - u'_2]_{pk_\Sigma}$.³

Then, CP and CSP jointly calculates $[u^*]_{pk_\Sigma} = \text{SMD}([u_1]_{pk_\Sigma}, [u_2]_{pk_\Sigma})$.

If $u^* = 1$, it indicates that the two keywords are the same. Otherwise, $u^* = 0$.

4 PROPOSED SYSTEM

In the proposed system (shown in Fig. 1), KGC first generates the master secret key for the system and the public/secret key pairs for the users, and partial strong secret keys for the CP and CSP (see step 1 in Fig. 1). Then, the data users apply search authorization from a single data owner or multiple data owners (step 2). The data owner sets an authorization time such that the search privilege expires automatically when the time is out. In the encryption phase (step 3), the data owner encrypts the file and a set of keywords, which are extracted from the file. To measure the importance of the keywords, a set of keyword weights are set and encrypted by the data owner. The encrypted files and keyword indexes are then uploaded to the CP. In the query phase (step 4), the data user designates a set of query keywords and a set of preference scores for the query

3. If $u_1 = 1$, it indicates $X \geq Y$; otherwise, $u_1 = 0$. If $u_2 = 1$, it indicates $Y \geq X$; otherwise, $u_2 = 0$.

TABLE 1
Notations

Notation	Description
MPK/MSK	master public/secret key
pk/sk	public/secret key of user
SK_1, SK_2	partial strong secret keys
$\mathcal{L}(X)$	bit length of X
$[X]_{pk}$	the ciphertext of X (encrypted by pk)
$SEnc/SDec$	symmetric encryption/decryption algorithm
$Sig/Verify$	signature/verification algorithm
CER/RVK	authorization/revocation certificate
AT/RT	authorization time period/revocation time
kw/α	extracted keyword/keyword weight
qw/β	query keyword/preference score
\mathbb{W}/\mathbb{Q}	encrypted index/encrypted query
$M/ID/K$	file/file identity/file encryption key
$[I]_{pk_\Sigma}$	encrypted relevance score
K2C	secure keyword to ciphertext algorithm
KET	secure keyword equivalent test protocol across domains
MKS	secure multiple keyword search protocol across domains
MAX	secure maximum protocol across domains
MAX_n	secure maximum out of n protocol across domains
TOP-k	secure top- k data retrieval protocol across domains

keywords. Then, he generates the trapdoor of the query keywords and preference scores. The trapdoor is sent to the CP to issue a search query. In the search phase, the CP verifies the search authority of the data user. If it is valid, CP calculates the relevance score of the encrypted file and the trapdoor. The calculated relevance score is in the encrypted form. Then, CP leverages the secure data retrieval protocol to get the top- k relevant encrypted files, which are returned to the data user (step 5). In the decryption phase, the data user recovers the top- k plaintext file (step 6). The communication channel in the system should be secure channel, protected using, for example, SSL (Secure Sockets Layer) or TLS (Transport Layer Security). Table 1 shows the main notations in this paper.

4.1 Key Generation

KGC runs *KeyGen* algorithm of PCTD (Section 3) to generate the system public parameter $PP = (g, N)$, strong master secret key $MSK = \lambda$ and user A_i 's public/secret key pair $pk_{A_i} = g^{\beta_i}, sk_{A_i} = \theta_i$. KGC also generates a master public key $MPK = g^\lambda$. MSK is kept secret by KGC and MPK is public. Then, KGC executes master secret key splitting algorithm of PCTD to generate partial strong secret keys $SK_1 = \lambda_1$ and $SK_2 = \lambda_2$, which are secretly sent to CP and CSP, respectively. sk_{A_i} is confidentially sent to user A_i and pk_{A_i} is public.

Let $SEnc/SDec$ be a symmetric encryption/decryption pair (with key space \mathcal{K}) that is cryptographically secure. Let $Sig/Verify$ be signature generation/verification pair that is strongly unforgeable. The concrete algorithms are not specified in this paper. We also define hash functions $H_1: \{0, 1\}^* \rightarrow Z_N$ and $H_2: Z_N \rightarrow \mathcal{K}$.

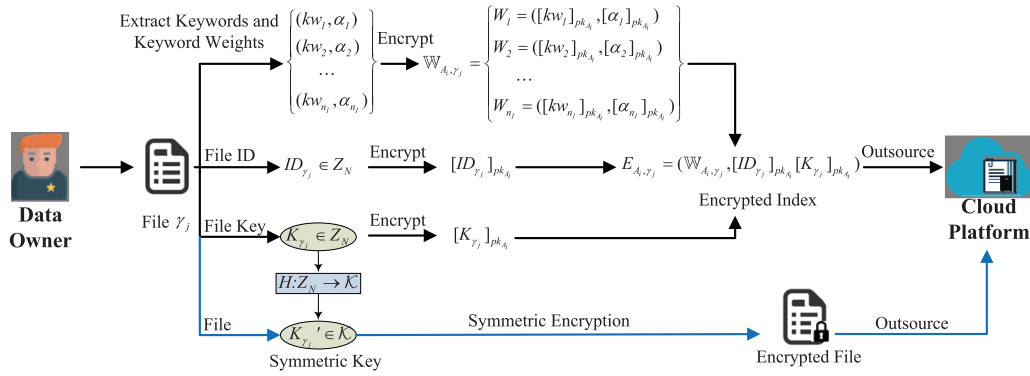


Fig. 3. Encryption illustration.

To simplify the presentation, we utilize the element in Z_N to be the secret key of *Sig* algorithm. In practical usage, the signature key is easily derived from the element in Z_N using a hash function.

4.2 User Authorization and Revocation

4.2.1 Single Data Owner Scenario

Suppose a user B wants to be authorized to search over data owner A_1 's data from 1st Jan. 2016 to 1st Jan. 2017 (authorization time $AT_1 = "20160101 - 20170101"$). The user B sends the message (B, AT_1) to data owner A_1 to apply for the authorization. If it is permitted, A_1 generates the authorization certificate for B .

$$CER_{A_1, B} = \langle cer = (A_1, B, AT_1, pk_{\Sigma}), Sig(cer, sk_{A_1}) \rangle,$$

where $pk_{\Sigma} = g^{sk_{\Sigma}}$, $sk_{\Sigma} = H_1(A_1, B, AT_1, sk_{A_1})$.

The secret key sk_{Σ} is given to B via secure channel. $CER_{A_1, B}$ is sent to KGC, CP, CSP and the user B . It is automatically invalid when the current time is out of AT_1 and user's authorization is revoked.

If A_1 plans to revoke B 's privilege in the period of AT_1 , A_1 creates a revocation certificate $RVK_{A_1, B}$ as

$$\langle rvk = (CER_{A_1, B}, revoke, RT), Sig(rvk, sk_{A_1}) \rangle,$$

where RT is the revocation time. Then, $RVK_{A_1, B}$ is sent to KGC, CP, CSP and user B .

4.2.2 Multiple Data Owners Scenario

Suppose a user B wants to simultaneous query information from data owners (A_1, \dots, A_m) 's files. He should first get the permissions $(CER_{A_i, B}, 1 \leq i \leq m)$ from each owner. Then, he applies the simultaneously query authorization from KGC. After verifying the certificates, KGC calculates the authorization time period $AT_{\Sigma} = AT_1 \cap \dots \cap AT_m$. Then KGC creates authorization certificate $CER_{\Sigma, B}$ as

$$\langle cer = (A_1, \dots, A_m, B, AT_{\Sigma}, pk_{\Sigma}), Sig(cer, MSK) \rangle,$$

where $pk_{\Sigma} = g^{sk_{\Sigma}}$, $sk_{\Sigma} = H_1(A_1, \dots, A_m, B, AT_{\Sigma}, MSK)$. sk_{Σ} is confidentially delivered to user B and pk_{Σ} is public to CP, CSP and user B .

To revoke $CER_{A_i, B}$ within the time period AT_{Σ} , KGC has to generate a revocation certificate $RVK_{\Sigma, B}$ as

$$\langle rvk = (CER_{\Sigma, B}, revoke, RT), Sig(rvk, MSK) \rangle,$$

where RT is the revocation time. Then, $RVK_{\Sigma, B}$ is sent to CP, CSP and user B .

4.3 Encryption

Suppose a data owner A_i wants to upload a file M_{γ_j} to the cloud. He follows the steps below to encrypt the data. Fig. 3 illustrates the encryption algorithm.

- 1) Data owner A_i extracts a set of keywords $\{kw_1, \dots, kw_{n_1}\}$ to describe the file. In order to differentiate the importance of the keyword, A_i sets keyword weights $\{\alpha_1, \dots, \alpha_{n_1}\} \in Z_N$ for the keywords. There are many ways to compute keyword weight, such as TF-IDF (term frequency and inverse document frequency). The data owner selects a method to define the keyword weight, which is not specified in this paper.
- 2) A_i encrypts the keywords using **K2C** algorithm to get $\{[kw_1]_{pk_{A_i}}, \dots, [kw_{n_1}]_{pk_{A_i}}\}$. The keyword weights are encrypted (using the PCTD algorithm) to $\{[\alpha_1]_{pk_{A_i}}, \dots, [\alpha_{n_1}]_{pk_{A_i}}\}$. A pair of encrypted keyword and keyword weight is denoted as $W_k = ([kw_k]_{pk_{A_i}}, [\alpha_k]_{pk_{A_i}})$, $1 \leq k \leq n_1$. Denote the set of encrypted keyword and keyword weight pair as $\mathbb{W}_{A_i, \gamma_j} = (W_1, \dots, W_{n_1})$.
- 3) The file identity $ID_{\gamma_j} \in Z_N$ and file encryption key $K_{\gamma_j} \in Z_N$ are encrypted (using the PCTD algorithm) to $[ID_{\gamma_j}]_{pk_{A_i}}$ and $[K_{\gamma_j}]_{pk_{A_i}}$.
- 4) Using hash function $H_2: Z_N \rightarrow \mathcal{K}$, the file encryption key K_{γ_j} is hashed to $K'_{\gamma_j} = H_2(K_{\gamma_j}) \in \mathcal{K}$. Then, A_i utilizes symmetric encryption algorithm *SEnc* to encrypt the file M_{γ_j} into an encrypted file $C_{\gamma_j} = SEnc(M_{\gamma_j}, K'_{\gamma_j})$.
- 5) A_i sends $E_{A_i, \gamma_j} = (\mathbb{W}_{A_i, \gamma_j}, [ID_{\gamma_j}]_{pk_{A_i}}, [K_{\gamma_j}]_{pk_{A_i}})$ and encrypted file C_{γ_j} to cloud platform.

Discussion. If the keyword weight is a decimal (such as TF-IDF), the data owner multiplies all the keyword weights with an integer (such as 10 or 100) such that these decimals are mapped into Z_N .

4.4 Query

In the query phase, the user B generates a trapdoor to issue the query (Fig. 4).

- 1) B specifies the query keywords $\{qw_1, \dots, qw_{n_2}\}$ and the preference scores of the keywords $\{\beta_1, \dots, \beta_{n_2}\}$, which indicates the importance of the keyword in the query.

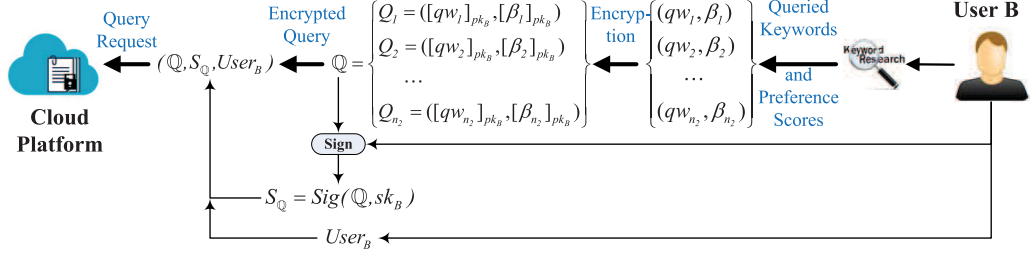


Fig. 4. Query illustration.

- 2) B uses **K2C** to encrypt the query keywords and obtains $\{[qw_1]_{pk_B}, \dots, [qw_{n_2}]_{pk_B}\}$. The preference scores are encrypted (using the PCTD algorithm) to $\{[\beta_1]_{pk_B}, \dots, [\beta_{n_2}]_{pk_B}\}$. Denote tuple $Q_k = \{[qw_k]_{pk_B}, [\beta_k]_{pk_B}\}$ ($1 \leq k \leq n_2$) and $\mathbb{Q} = (Q_1, \dots, Q_{n_2})$.
- 3) B signs on the query \mathbb{Q} using his private key sk_B and generates a signature $S_{\mathbb{Q}} = \text{Sig}(\mathbb{Q}, sk_B)$.
- 4) B sends the encrypted query \mathbb{Q} , signature $S_{\mathbb{Q}}$ and his identity $User_B$ to cloud platform.

4.5 Search

Receiving the keyword search request, cloud platform first verifies whether B is authorized to access to the data. If B has the privilege, CS verifies the signature $S_{\mathbb{Q}}$ of encrypted query \mathbb{Q} using the user B 's public key pk_B . If it is not valid, the query is rejected. Otherwise, CP responds to the search query as follows. (1) CP first computes the relevance score of the encrypted query \mathbb{Q} and the encrypted index for each authorized document. (2) Then, the encrypted files that have the top- k highest relevance scores are returned to user. The concrete procedures are illustrated as below.

4.5.1 Relevance Score Computation

In order to calculate the relevance score between search query and the file index, we design a secure multiple keyword search protocol across domains (**MKS**) to compute the score.

Algorithm 1. Secure Multiple Keyword Search Protocol Across Domains (**MKS**)

Input: Encrypted indexes $\mathbb{W} = (W_1, \dots, W_{n_1})$ and search query $\mathbb{Q} = (Q_1, \dots, Q_{n_2})$, where $W_i = \langle [X_i]_{pk_A}, [\alpha_i]_{pk_A} \rangle$, $Q_j = \langle [Y_j]_{pk_B}, [\beta_j]_{pk_B} \rangle$.

Output: $[I]_{pk_{\Sigma}}$.

- 1 Initialize $[I]_{pk_{\Sigma}} = [0]_{pk_{\Sigma}}$;
- 2 **for** $j = 1$ to n_2 **do**
- 3 **for** $i = 1$ to n_1 **do**
- 4 CP and CSP jointly calculate $[u_i]_{pk_{\Sigma}} = \text{KET}([X_i]_{pk_A}, [Y_j]_{pk_B})$;
- 5 $[s'_i]_{pk_{\Sigma}} = \text{SMD}([\alpha_i]_{pk_A}, [\beta_j]_{pk_B})$;
- 6 $[s_i]_{pk_{\Sigma}} = \text{SMD}([u_i]_{pk_{\Sigma}}, [s'_i]_{pk_{\Sigma}})$;
- 7 CP calculates $[I]_{pk_{\Sigma}} = [I]_{pk_{\Sigma}} \cdot [s_i]_{pk_{\Sigma}}$;
- 8 **Return** $[I]_{pk_{\Sigma}}$.

The inputs of **MKS** protocol are the encrypted index $\mathbb{W}_{A_i, \gamma_j} = (W_1, \dots, W_{n_1})$ and the encrypted query $\mathbb{Q} = (Q_1, \dots, Q_{n_2})$, where $W_{k_1} = \langle [kw_{k_1}]_{pk_{A_i}}, [\alpha_{k_1}]_{pk_{A_i}} \rangle$ for $1 \leq k_1 \leq n_1$ and $Q_{k_2} = \langle [qw_{k_2}]_{pk_B}, [\beta_{k_2}]_{pk_B} \rangle$ for $1 \leq k_2 \leq n_2$. It outputs an encrypted relevance score $[I]_{pk_{\Sigma}}$ under public key pk_{Σ} .

For each queried keyword Y_j ($1 \leq j \leq n_2$ in Line 2), the **MKS** protocol calculates its relevance score with the encrypted index. In order to achieve this purpose, the algorithm first computes the relevance score of Y_j and X_i ($1 \leq i \leq n_1$ in Line 3).

- 1) In line 4, it utilizes **KET** algorithm to test whether $X_i = Y_j$. It outputs $[u_i]_{pk_{\Sigma}} = [1]_{pk_{\Sigma}}$ if $X_i = Y_j$. Otherwise, it outputs $[u_i]_{pk_{\Sigma}} = [0]_{pk_{\Sigma}}$.
- 2) In line 5, it calculates the multiplication of the keyword weight α_i (of X_i) and the preference score β_j (of Y_j)

$$[s'_i]_{pk_{\Sigma}} = [\alpha_i \cdot \beta_j]_{pk_{\Sigma}} = \text{SMD}([\alpha_i]_{pk_A}, [\beta_j]_{pk_B}).$$

- 3) In line 6, if $X_i = Y_j$, the relevance score of X_i and Y_j is $[s_i]_{pk_{\Sigma}} = [\alpha_i \cdot \beta_j]_{pk_{\Sigma}}$ since

$$\text{SMD}([u_i]_{pk_{\Sigma}}, [s'_i]_{pk_{\Sigma}}) = \text{SMD}([1]_{pk_{\Sigma}}, [\alpha_i \cdot \beta_j]_{pk_{\Sigma}}).$$

Otherwise, $[s_i]_{pk_{\Sigma}} = [0]_{pk_{\Sigma}}$ since

$$\text{SMD}([u_i]_{pk_{\Sigma}}, [s'_i]_{pk_{\Sigma}}) = \text{SMD}([0]_{pk_{\Sigma}}, [\alpha_i \cdot \beta_j]_{pk_{\Sigma}}).$$

- 4) In line 7, the relevance score s_i is added to I

$$[I]_{pk_{\Sigma}} \cdot [s_i]_{pk_{\Sigma}} = [I + s_i]_{pk_{\Sigma}}.$$

After the computation of relevance score, $E_{A_i, \gamma_j} = (\mathbb{W}_{A_i, \gamma_j}, [ID_{\gamma_j}]_{pk_{A_i}}, [K_{\gamma_j}]_{pk_{A_i}})$ is transformed into $T_{A_i, \gamma_j} = ([I_{\gamma_j}]_{pk_{\Sigma}}, [ID_{\gamma_j}]_{pk_{A_i}}, [K_{\gamma_j}]_{pk_{A_i}})$ (Fig. 5).

Note: The computation overhead of the **MKS** protocol does not linearly increase with $n_1 \times n_2$. The keyword equality tests executed by the **KET** protocol are independent and can be calculated in parallel. The computation time of **MKS** protocol is a little higher than that of the **KET** protocol when parallel computation is leveraged, rather than linearly grow with $n_1 \times n_2$. It is demonstrated by the performance test in Section 5.

4.5.2 Top- k Ranking

After the relevance scores are calculated, it is necessary to find the top- k relevant encrypted files according to the scores. The requirements of the top- k ranking are listed

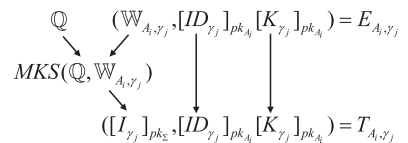


Fig. 5. Relevance score computation.

below. (1) During the ranking process, the plaintext of the encrypted relevance score should not be revealed to both CP and CSP. (2) The CP and CSP do not know which top- k documents are returned to user.

In order to realize top- k ranking, three privacy-preserving protocols are proposed.

- 1) Secure maximum protocol across domains (**MAX**) figures out the encrypted file (with the maximum relevance score) from two encrypted files.
- 2) Secure maximum out of n protocol across domains (**MAX_n**) utilizes **MAX** protocol to find the encrypted file (with the maximum relevance score) from n encrypted files.
- 3) Secure top- k data retrieval protocol across domains (**TOP- k**) leverages **MAX_n** protocol to find the top- k relevant files.

Secure Maximum Protocol Across Domains (MAX).

Given $T_{A_{i_1}, \gamma_{j_1}} = ([I_{\gamma_{j_1}}]_{pk_{\Sigma}}, [ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, [K_{\gamma_{j_1}}]_{pk_{A_{i_1}}})$ and $T_{A_{i_2}, \gamma_{j_2}} = ([I_{\gamma_{j_2}}]_{pk_{\Sigma}}, [ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, [K_{\gamma_{j_2}}]_{pk_{A_{i_2}}})$ that are encrypted using different keys, **MAX** protocol outputs a new tuple $T_U = ([I_U]_{pk_{\Sigma}}, [ID_U]_{pk_{\Sigma}}, [K_U]_{pk_{\Sigma}})$, such that $I_U = \max(I_{\gamma_{j_1}}, I_{\gamma_{j_2}})$ and ID_U, K_U correspond to its file identity and file encryption key. In the protocol, CP and CSP could not distinguish the source of T_U . It has three steps and requires the interaction between CP and CSP. (The correctness of **MAX** protocol is verified in Supplemental Materials B, available online.)

Step 1. CP computes:

$$\begin{aligned} [I'_{\gamma_{j_1}}]_{pk_{\Sigma}} &= [I_{\gamma_{j_1}}]_{pk_{\Sigma}}^2 \cdot [1]_{pk_{\Sigma}} = [2I_{\gamma_{j_1}} + 1]_{pk_{\Sigma}}; \\ [I'_{\gamma_{j_2}}]_{pk_{\Sigma}} &= [I_{\gamma_{j_2}}]_{pk_{\Sigma}}^2 = [2I_{\gamma_{j_2}}]_{pk_{\Sigma}}. \end{aligned}$$

The purpose of the calculation is that if $I_{\gamma_{j_1}} \geq I_{\gamma_{j_2}}$ and $I_{\gamma_{j_1}}, I_{\gamma_{j_2}} \geq 0$, we have $I'_{\gamma_{j_1}} > I'_{\gamma_{j_2}}$.

Then, CP randomly selects $r_1, r'_1, r_2, r_3, r_4 \in \mathbb{Z}_N$, where $\mathcal{L}(r_1) < \mathcal{L}(N)/4 - 1$, $\mathcal{L}(r'_1) < \mathcal{L}(N)/8$. Then, CP flips a random coin $s \in \{0, 1\}$.

If $s = 1$, CP and CSP jointly calculate:

$$\begin{aligned} C_1 &= ([I'_{\gamma_{j_1}}]_{pk_{\Sigma}})^{r_1} \cdot ([I'_{\gamma_{j_2}}]_{pk_{\Sigma}})^{N-r_1} \cdot [r'_1]_{pk_{\Sigma}} \\ &= [r_1(I'_{\gamma_{j_1}} - I'_{\gamma_{j_2}}) + r'_1]_{pk_{\Sigma}}; \end{aligned} \quad (1)$$

$$\begin{aligned} C_2 &= [I_{\gamma_{j_2}}]_{pk_{\Sigma}} \cdot ([I_{\gamma_{j_1}}]_{pk_{\Sigma}})^{N-1} \cdot [r_2]_{pk_{\Sigma}} \\ &= [I_{\gamma_{j_2}} - I_{\gamma_{j_1}} + r_2]_{pk_{\Sigma}}; \end{aligned} \quad (2)$$

$$\begin{aligned} C_3 &= \text{SAD}([ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, ([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}})^{N-1}) \cdot [r_3]_{pk_{\Sigma}} \\ &= [ID_{\gamma_{j_2}} - ID_{\gamma_{j_1}} + r_3]_{pk_{\Sigma}}; \end{aligned} \quad (3)$$

$$\begin{aligned} C_4 &= \text{SAD}([K_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, ([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}})^{N-1}) \cdot [r_4]_{pk_{\Sigma}} \\ &= [K_{\gamma_{j_2}} - K_{\gamma_{j_1}} + r_4]_{pk_{\Sigma}}. \end{aligned} \quad (4)$$

If $s = 0$, CP and CSP jointly calculates:

$$\begin{aligned} C_1 &= ([I'_{\gamma_{j_2}}]_{pk_{\Sigma}})^{r_1} \cdot ([I'_{\gamma_{j_1}}]_{pk_{\Sigma}})^{N-r_1} \cdot [r'_1]_{pk_{\Sigma}} \\ &= [r_1(I'_{\gamma_{j_2}} - I'_{\gamma_{j_1}}) + r'_1]_{pk_{\Sigma}}; \end{aligned} \quad (5)$$

$$\begin{aligned} C_2 &= [I_{\gamma_{j_1}}]_{pk_{\Sigma}} \cdot ([I_{\gamma_{j_2}}]_{pk_{\Sigma}})^{N-1} \cdot [r_2]_{pk_{\Sigma}} \\ &= [I_{\gamma_{j_1}} - I_{\gamma_{j_2}} + r_2]_{pk_{\Sigma}}; \end{aligned} \quad (6)$$

$$\begin{aligned} C_3 &= \text{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, ([ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}})^{N-1}) \cdot [r_3]_{pk_{\Sigma}} \\ &= [ID_{\gamma_{j_1}} - ID_{\gamma_{j_2}} + r_3]_{pk_{\Sigma}}; \end{aligned} \quad (7)$$

$$\begin{aligned} C_4 &= \text{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, ([K_{\gamma_{j_2}}]_{pk_{A_{i_2}}})^{N-1}) \cdot [r_4]_{pk_{\Sigma}} \\ &= [K_{\gamma_{j_1}} - K_{\gamma_{j_2}} + r_4]_{pk_{\Sigma}}. \end{aligned} \quad (8)$$

CP utilizes partial strong secret key SK_1 to calculate $C'_1 = \text{PD}_{1, SK_1}(C_1)$ and sends C'_1, C_1, C_2, C_3, C_4 to CSP.

Step 2. After receiving C'_1, C_1, C_2, C_3, C_4 , CSP calculates C''_1 by using his partial strong secret key SK_2

$$C''_1 = \text{PD}_{2, SK_2}(C_1, C'_1).$$

If $C''_1 < \mathcal{L}(N)/2$, CSP sets $\alpha = 0$ and computes

$$C_5 = [0]_{pk_{\Sigma}}, C_6 = [0]_{pk_{\Sigma}}, C_7 = [0]_{pk_{\Sigma}}.$$

If $C''_1 > \mathcal{L}(N)/2$, CSP sets $\alpha = 1$ and computes

$$C_5 = \text{CR}(C_2), C_6 = \text{CR}(C_3), C_7 = \text{CR}(C_4).$$

Then, CSP encrypts $[\alpha]_{pk_{\Sigma}}$ and sends $([\alpha]_{pk_{\Sigma}}, C_5, C_6, C_7)$ to CP.

Step 3. When $([\alpha]_{pk_{\Sigma}}, C_5, C_6, C_7)$ is received, CP executes the following computations according to the value of s tossed in step 1.

If $s = 1$, CP and CSP jointly compute

$$\begin{aligned} [I_U]_{pk_{\Sigma}} &= [I_{\gamma_{j_1}}]_{pk_{\Sigma}} \cdot C_5 \cdot ([\alpha]_{pk_{\Sigma}})^{N-r_2}; \\ [ID_U]_{pk_{\Sigma}} &= \text{SAD}([ID_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_6) \cdot ([\alpha]_{pk_{\Sigma}})^{N-r_3}; \\ [K_U]_{pk_{\Sigma}} &= \text{SAD}([K_{\gamma_{j_1}}]_{pk_{A_{i_1}}}, C_7) \cdot ([\alpha]_{pk_{\Sigma}})^{N-r_4}. \end{aligned}$$

If $s = 0$, CP and CSP jointly compute

$$\begin{aligned} [I_U]_{pk_{\Sigma}} &= [I_{\gamma_{j_2}}]_{pk_{\Sigma}} \cdot C_5 \cdot ([\alpha]_{pk_{\Sigma}})^{N-r_2}; \\ [ID_U]_{pk_{\Sigma}} &= \text{SAD}([ID_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, C_6) \cdot ([\alpha]_{pk_{\Sigma}})^{N-r_3}; \\ [K_U]_{pk_{\Sigma}} &= \text{SAD}([K_{\gamma_{j_2}}]_{pk_{A_{i_2}}}, C_7) \cdot ([\alpha]_{pk_{\Sigma}})^{N-r_4}. \end{aligned}$$

It is obvious that I_U is the maximum relevance score between $I_{\gamma_{j_1}}$ and $I_{\gamma_{j_2}}$.

Secure Maximum Out of n Protocol Across Domains (MAX_n). Taken as input n encrypted tuples T_1, \dots, T_n , the **MAX_n** protocol outputs a new tuple

$$T_{MAX} = ([I_{MAX}]_{pk_{\Sigma}}, [ID_{MAX}]_{pk_{\Sigma}}, [K_{MAX}]_{pk_{\Sigma}}),$$

such that $I_{MAX} = \max(I_1, \dots, I_n)$ and ID_{MAX}, K_{MAX} correspond to its file identity and file encryption key. In the protocol, CP and CSP cannot recover or distinguish the source of T_{MAX} . The construction is illustrated as below.

As shown in Fig. 6, **MAX_n** protocol needs $\lceil \log_2 n \rceil$ rounds to find the maximum tuple. In each round, the maximum tuple of the two adjacent encrypted tuples is computed (use **MAX** protocol). After $\lceil \log_2 n \rceil$ rounds, there is only one tuple left, which is the maximum tuple T_{MAX} .

Secure Top- k Data Retrieval Protocol Across Domains (TOP- k). On input n encrypted tuples T_1, \dots, T_n , **TOP- k**

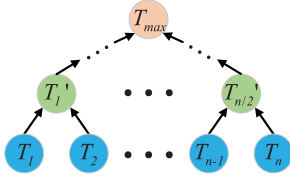


Fig. 6. Illustration of MAX_n protocol.

protocol outputs the k new encrypted tuples that have the k highest relevance scores. The protocol is illustrated as following.

First, it initializes an empty set S_a to store the top- k result. The set S_b is assigned with T_1, \dots, T_n . The **TOP- k** protocol needs k rounds to get the result. In each round, the protocol picks up the maximum tuple. The round is illustrated in detail as following.

- 1) (Line 3-4) Run MAX_n protocol to get the maximum tuple T_{MAX_i} in the i th round, which is inserted into S_a .
- 2) (Line 5-7) For each encrypted tuple in S_b , CP and CSP calculates

$$\begin{aligned} V_j &= \text{SAD}([ID_{\text{MAX}_i}]_{pk_\Sigma})^{r_j}, (T_{j,2})^{(N-r_j)} \\ &= [r_j(ID_{\text{MAX}_i} - ID_j)]_{pk_\Sigma}. \end{aligned}$$

If $ID_j = ID_{\text{MAX}_i}$, we have $V_j = [0]_{pk_\Sigma}$. Otherwise, $V_j \neq [0]_{pk_\Sigma}$. Then, CP partially decrypts V_j and stores the result in V'_j .

- 3) (Line 8) To conceal the plaintext information, CP uses a function π_i to permute (V_1, \dots, V_n) and (V'_1, \dots, V'_n) . Then, $\{(V_{\pi_i(j)}, V'_{\pi_i(j)})\}$ for $1 \leq j \leq n$ are sent to CSP.
- 4) (Line 9-14) CSP decrypts $\{(V_{\pi_i(j)}, V'_{\pi_i(j)})\}$ to get β_j for $1 \leq j \leq n$. If $\beta_j = 0$, CSP sets $A_{\pi_i(j)} = [0]_{pk_\Sigma}$. Otherwise, $A_{\pi_i(j)} = [1]_{pk_\Sigma}$.
- 5) (Line 15) Receiving $(A_{\pi_i(1)}, \dots, A_{\pi_i(n)})$, CP first utilizes π_i^{-1} to recover the order and obtains (A_1, \dots, A_n) . It is easy to find that the origin tuple T_ζ of T_{MAX_i} has $A_\zeta = [0]_{pk_\Sigma}$ and other tuples have $A_j = [1]_{pk_\Sigma}$ for $1 \leq j \leq n$ and $j \neq \zeta$.
- 6) (Line 16-18) The $\{[I_1]_{pk_\Sigma}, \dots, [I_n]_{pk_\Sigma}\} \in S_b$ are refreshed. $[I_\zeta]_{pk_\Sigma}$ in the origin tuple T_ζ of T_{MAX_i} are set to be $[0]_{pk_\Sigma}$ since $[I_\zeta]_{pk_\Sigma} = \text{SMD}(T_{\zeta,1}, A_j) = \text{SMD}([I_\zeta]_{pk_\Sigma}, [0]_{pk_\Sigma}) = [I_\zeta * 0]_{pk_\Sigma} = [0]_{pk_\Sigma}$.

For $1 \leq j \leq n$ and $j \neq \zeta$, $[I_j]_{pk_\Sigma}$ are changed since

$$[I_j]_{pk_\Sigma} = \text{SMD}(T_{j,1}, A_j) = \text{SMD}([I_j]_{pk_\Sigma}, [1]_{pk_\Sigma}) = [I_j * 1]_{pk_\Sigma} = [I_j]_{pk_\Sigma}.$$

After k rounds computation, S_a contains k tuples that have the k highest relevance scores.

4.6 Decryption

Receiving the top- k encrypted files, the user B utilizes public key pk_Σ to recover the relevance score I_i , file identity ID_i and the key K_i for $1 \leq i \leq k$. Then, using the private information retrieval (PIR) [49], [50] or oblivious RAM (ORAM) methods [51], [52], the user securely gets back the corresponding encrypted files from CP without leaking the access pattern. The key K_i is hashed to symmetric key $K'_i = H_2(K_i) \in \mathcal{K}$, which is utilized to decrypt the file M_i .

Algorithm 2. Secure Top- k Data Retrieval Protocol Across Domains (**TOP- k**)

Input: CP has n ciphertext T_1, \dots, T_n , ($k < n$), where $T_i = \langle T_{i,1}, T_{i,2}, T_{i,3} \rangle = \langle [I_i]_{pk_\Sigma}, [ID_i]_{pk_{A_i}}, [K_i]_{pk_{A_i}} \rangle$.

Output: CP gets top- k file names and file secret keys (in encrypted form) corresponding to top- k relevance scores.

- 1 Initialize sets $S_a = \emptyset$ and $S_b = \{T_1, \dots, T_n\}$;
- 2 **for** $i = 1$ to k **do**
- 3 CP and CSP jointly run $T_{\text{MAX}_i} = \text{MAX}_n(T_1, \dots, T_n)$ to get the tuple $T_{\text{MAX}_i} = \langle [I_{\text{MAX}_i}]_{pk_\Sigma}, [ID_{\text{MAX}_i}]_{pk_\Sigma}, [K_{\text{MAX}_i}]_{pk_\Sigma} \rangle$ with maximum relevance score, where $T_1, \dots, T_n \in S_b$;
- 4 Insert the tuple T_{MAX_i} into set S_a ;
- 5 **for** $j = 1$ to n **do**
- 6 CP and CSP jointly calculate: $V_j = \text{SAD}([ID_{\text{MAX}_i}]_{pk_\Sigma})^{r_j}, (T_{j,2})^{(N-r_j)}$, in which $r_j \in \mathbb{Z}_N$ is a random number;
- 7 (@CP): Partially decrypt V_j to $V'_j = \text{PD1}_{SK_1}(V_j)$;
- 8 Permute (V_j, V'_j) using permutation function π_i . The result is denoted as $(V_{\pi_i(j)}, V'_{\pi_i(j)})$, which are sent to CSP;
- 9 (@CSP): Decrypt $V_{\pi_i(j)} = \text{PD2}_{SK_2}(V_{\pi_i(j)}, V'_{\pi_i(j)})$ by using SK_2 and denote it as $\beta_j = V''_{\pi_i(j)}$;
- 10 **if** $\beta_j = 0$ **then**
- 11 denote $A_{\pi_i(j)} = [0]_{pk_\Sigma}$;
- 12 **else**
- 13 denote $A_{\pi_i(j)} = [1]_{pk_\Sigma}$;
- 14 Send $A_{\pi_i(j)}$ back to CP;
- 15 (@CP) CP obtains A_j by using permutation π_i^{-1} ;
- 16 CP refreshes $\{[I_1]_{pk_\Sigma}, \dots, [I_n]_{pk_\Sigma}\} \in S_b$ by computing
- 17 **for** $j = 1$ to n **do**
- 18 CP and CSP jointly calculate: $[I_j]_{pk_\Sigma} = \text{SMD}(T_{j,1}, A_j)$;
- 19 **Return** the set $S_a = (T_{\text{MAX}_1}, \dots, T_{\text{MAX}_k})$.

4.7 Implication of the System

This system can be used in the secure storage system, such as the secure cloud storage system, encrypted healthcare record storage system, secure multimedia storage system, etc. Take the secure electronic health storage system as an example. The patients are the data owners, and the doctors and nurses are the data users. Suppose that a medical director A of the pediatric department has got the access authorization from a set of young patients (B_1, B_2, B_3) . The young patient B_1 suffered from pedopneumonia with a high fever. The keywords extracted from his electronic health record (EHR) (file identity ID_1) are "pedopneumonia, fever" with keywords weights "7, 3". B_2 suffered from parascarlantina with a low fever and a slight cough, and his EHR (file identity ID_2) has keywords "parascarlantina, fever, cough" with weights "7, 1, 1". B_3 suffered from pedopneumonia with a severe cough, and his EHR (file identity ID_3) has keywords "pedopneumonia, cough" with weights "7, 5". These files and keywords and weights are encrypted using the encryption algorithm in Section 4.3.

The medical director A studies the pedopneumonia disease. He figures out a set of query keywords "pedopneumonia, fever, cough" with preference scores "5, 2, 1", and wants to get the top-2 results. The search query is encrypted using the query algorithm in Section 4.4. Then, CP executes the search algorithm in Section 4.5 and returns the encrypted EHRs with file identities ID_1, ID_3 . In the search process, the CP and CSP do not know which encrypted files match with

TABLE 2
K2C Execution Time (s)

$\mathcal{L}(N)$	512	768	1024	1280	1536	1792	2048
K2C	0.005	0.007	0.016	0.029	0.055	0.074	0.110

TABLE 3
KET Execution Time (s)

$\mathcal{L}(N)$	512	768	1024	1280	1536	1792	2048
CP	0.049	0.162	0.333	0.671	1.036	1.637	2.450
CSP	0.005	0.019	0.039	0.078	0.126	0.198	0.297
Total	0.054	0.181	0.372	0.749	1.162	1.835	2.747

TABLE 4
MAX Execution Time (s)

$\mathcal{L}(N)$	512	768	1024	1280	1536	1792	2048
CP	0.059	0.201	0.360	0.723	1.319	2.026	2.729
CSP	0.007	0.023	0.042	0.084	0.160	0.244	0.330
Total	0.066	0.224	0.402	0.807	1.479	2.270	3.059

the query trapdoor and which documents are returned to the user. Finally, A decrypts the result using the authorization secret key sk_{Σ} .

5 PERFORMANCE ANALYSIS

We conduct extensive experiments to evaluate the performance of our MRSE in this section. The experiments are performed on PC running Windows 7 64-bit operation system with the following settings: Intel(R) Core(TM) i7-4790 CPU @3.60 GHz, 12 GB RAM. All the protocols and algorithms are programmed using Java language and executed by Eclipse application program. We utilize multi-thread programming method to implement the system.

The length of N (denoted as $\mathcal{L}(N)$) affects the performance of the protocols to a great extent. Both the running time and communication cost of these protocols increase with the length of N . We utilize 512, 768, 1024, 1280, 1536, 1792 and 2048 bits to be $\mathcal{L}(N)$ in the following experiments, respectively. When $\mathcal{L}(N) = 1024$, it achieves 80-bit security level [53]. We recommend the readers to use $\mathcal{L}(N) = 1024$ as the parameter in real applications to balance the performance and security level.

We set the file number $|F| = 128, 512, 1024, 2048, 4096$ to test the system performance. In the test, eight keywords are extracted from a file to build the encrypted index and four keyword are specified to generate a keyword trapdoor. The number of the extracted keyword and the queried keyword can be changed according to the real application. In the *Encryption* and *Query* algorithms, we also specify other keyword numbers to evaluate their performances.

5.1 Performance of Crypto Primitives and Protocols

First, we test the performance of the basic algorithms and protocols of our proposed system on the PC in terms of computation and communication overhead. Shown in Table 2, 3, 4, 5, and 6, all these protocols have performances increase with $\mathcal{L}(N)$. The reason is that the overhead of basic

TABLE 5
MKS Execution Time (s)

$\mathcal{L}(N)$	512	768	1024	1280	1536	1792	2048
CP	0.213	0.496	0.981	1.836	3.473	4.418	6.807
CSP	0.025	0.059	0.118	0.221	0.451	0.533	0.812
Total	0.238	0.555	1.099	2.057	3.888	4.951	7.619

TABLE 6
Communication Overhead (KB)

$\mathcal{L}(N)$	512	768	1024	1280	1536	1792	2048
KET	4.33	6.51	8.69	10.86	13.04	15.21	17.40
MAX	1.02	1.53	2.04	2.55	3.06	3.58	4.09
MKS	78.2	117.2	156.4	195.6	234.7	274.0	313.0

TABLE 7
MAX_n Execution Time (s)

$ F $	128	512	1024	2048	4096
CP	2.520	3.240	3.602	3.865	4.215
CSP	0.294	0.377	0.421	0.557	0.749
Total	2.814	3.617	4.023	4.422	4.964

calculations (such as modular addition, multiplication and exponentiation) grow with $\mathcal{L}(N)$.

- Table 2 shows the running time of **K2C** algorithm. When $\mathcal{L}(N) = 1024$, **K2C** algorithm requires 0.016 s to encrypt a keyword into a ciphertext. Since this algorithm is executed by CP, there is no communication cost.
- Table 3 shows the computation cost of **KET** protocol. The communication cost between CP and CSP in **KET** protocol is depicted in Table 6. When $\mathcal{L}(N) = 1024$, $time_{CP} = 0.333$ s, $time_{CSP} = 0.039$ s, $time_{Total} = 0.372$ s and the communication cost is 8.69 KB. It indicates that 0.372 s running time and 8.69 KB data transmission is required for a privacy-preserving keyword equality test across domains.
- Table 4 shows the running time of **MAX** protocol that varies with the bit number of N . The transmission cost is shown in Table 6. When $\mathcal{L}(N) = 1024$, $time_{CP} = 0.360$ s, $time_{CSP} = 0.042$ s, $time_{Total} = 0.402$ s and the communication cost is 2.04 KB.
- Tables 5 and 6 describe the computation and communication overheads of **MKS** protocol, in which a set of **KET** protocols are executed to test the keywords equality. These equality tests can be paralleled executed to improve the efficiency using multi-thread programming method. In the test, we set the encrypted index with eight keywords and the search query with four keywords. When $\mathcal{L}(N) = 1024$, $time_{CP} = 0.981$ s, $time_{CSP} = 0.118$ s, $time_{Total} = 1.099$ s and the communication cost is 156.4 KB.
- Table 7 indicates that the computation cost of **MAX_n** protocol increases with the number n of encrypted files. In this table, $\mathcal{L}(N) = 1024$. As shown in Fig. 6, the **MAX** sub-protocol in the same level can be executed in parallel. If $|F| = n = 2^{\delta}$, the time consumed

TABLE 8
TOP-1 Execution Time (s)

$ F $	128	512	1024	2048	4096
CP	3.167	3.803	4.356	5.054	5.629
CSP	0.368	0.538	0.889	0.995	1.124
Total	3.535	4.341	5.245	6.049	6.753

TABLE 9
Communication Overhead (MB)

$ F $	128	512	1024	2048	4096
MAX _n	0.257	1.042	2.089	4.182	8.369
TOP-1	1.108	4.445	8.894	17.792	35.589

TABLE 10
Enc or Query Execution Time (s)

$ KW $	1	2	3	4	5
Total	0.032	0.048	0.064	0.080	0.096
$ KW $	6	7	8	9	10
Total	0.112	0.128	0.144	0.160	0.176

by MAX_n protocol approximately equals to δ times of the running time of MAX sub-protocol. With the parallel computation, when $|F| = 4096$, $time_{CP} = 5.629$ s, $time_{CSP} = 1.124$ s, $time_{Total} = 6.753$ s and the communication cost is 8.369 MB (shown in Table 9).

- Table 8 shows the computation overhead of TOP-1 protocol that varies with the number $|F|$ of encrypted files. Here we set $\mathcal{L}(N) = 1024$. The most important sub-protocol in TOP-1 is the MAX_n protocol. Similarly, we also use parallel computations. When $|F| = n = 4096$, $time_{CP} = 5.353$ s, $time_{CSP} = 1.885$ s, $time_{Total} = 7.238$ s and the communication cost of TOP-1 is 35.589 MB (shown in Table 9). Since each loop in TOP- k protocol is exactly the same. We only test the performance of TOP-1. The readers can easily deduce the performance of TOP- k by multiplying a factor k .

5.2 Performance of Multi-User MRSE

In this section, we test the performance of the suggested novel MRSE system. To achieve 80-bit security level, we choose the parameter $\mathcal{L}(N) = 1024$. The encrypted file number $|F| = 4096$.

- Since the encryption index generation process in *Encryption* algorithm and query generation process in *Query* algorithm are almost the same, the performance of these two algorithms are described in Table 10. The running time of *Encryption* and *Query* algorithms grows with the number of keywords (denoted as $|KW|$). When $|KW| = 6$ and 10, the execution times are 0.112 s and 0.176 s, respectively.
- The *Search* algorithm is executed by the interaction between CP and CSP. The computation time of *Search* algorithm (shown in Table 11) varies with the number of the returned results. If only one result is returned, $time_{CP} = 5.353$ s, $time_{CSP} = 1.885$ s, $time_{Total} = 7.238$ s. If five results are returned,

TABLE 11
Search Execution Time (s)

k	1	2	3	4	5
CP	5.353	10.706	16.059	21.411	26.765
CSP	1.885	3.769	5.653	7.537	9.422
Total	7.238	14.475	21.712	28.948	36.187

TABLE 12
KNN-SE Execution Time (s)

$ KW $	KeyGen	BuildIndex	Trapdoor	Relevance Score Calculation
500	0.017	0.008	0.390	0.003
1000	0.078	0.030	3.253	0.003
1500	0.136	0.110	11.398	0.003
2000	0.303	0.189	29.229	0.003
2500	0.379	0.442	56.778	0.003
3000	0.665	0.592	90.278	0.003
3500	0.730	0.781	158.527	0.003
4000	1.453	0.832	229.628	0.003
4500	1.608	1.069	326.443	0.003
5000	1.871	1.969	435.608	0.003
5500	2.115	2.341	585.812	0.003
6000	2.541	2.710	766.110	0.003

$time_{CP} = 26.765$ s, $time_{CSP} = 9.422$ s, $time_{Total} = 36.187$ s. In order to improve user's experience, it is important to reduce the wait time of data user after he submits a search query. One optimization method is to return only one result to user each time. When the user is decrypting the received result, the following results are continually returned to user.

The execution time of KNN-SE is shown in Table 12. KNN-SE has execution times rapidly increase with the size of predefined keywords set, which is denoted as $|KW|$. When $|KW| = 6000$, $KeyGen_{time} = 2.541$ s, $BuildIndex_{time} = 2.710$ s, $Trapdoor_{time} = 766.11$ s. When $|KW|$ becomes larger, the dimensions of the matrixes M_1 and M_2 increase, and the time to compute M_1^{-1} and M_2^{-1} grows quickly. The data users have to consume more time to generate a search query.

The computation overhead of our system is compared with KNN-SE in Fig. 7.

- In the key generation procedure (shown in Fig. 7a), our system requires 4×10^{-5} s to generate the master secret key and a public/secret key pair for the user. The key generation time of KNN-SE drastically increases with the size $|KW|$ of the predefined keywords set. When $|KW| = 500$, the key generation time of KNN-SE is 0.017 s; and that is 2.541 s when $|KW| = 6000$.
- In the encryption procedure (shown in Fig. 7b), assume that ten keywords are extracted from the file. Our system requires 0.176 s to build the encrypted keyword index. The build index time of KNN-SE quickly grows with $|KW|$. When $|KW| = 2000$, the build index time of KNN-SE is 0.189 s; and that is 2.710 s when $|KW| = 6000$.
- In the query procedure (shown in Fig. 7b), assume that ten keywords are queried. Our system requires 0.176 s to generate a trapdoor. The trapdoor

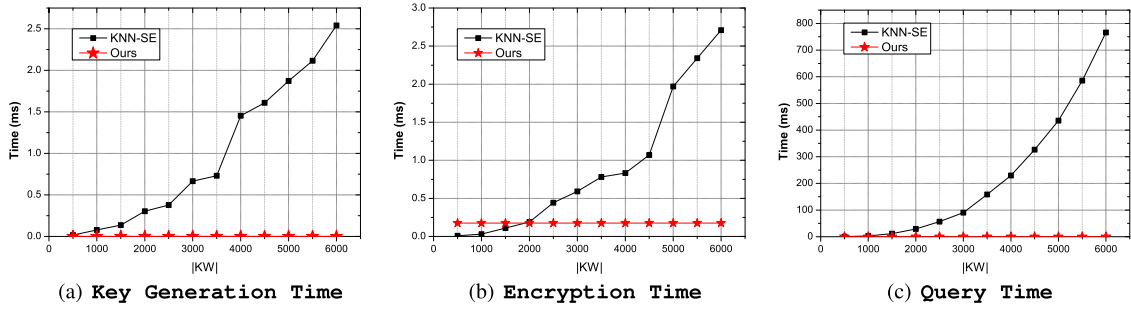


Fig. 7. Computation overhead comparison.

generation time of KNN-SE also rapidly increases with $|KW|$. When $|KW| = 500$, the build index time of KNN-SE is 0.390 s; and that is 766.110 s when $|KW| = 6000$.

- The search time is not plotted since that of KNN-SE is constant to be 0.003 s and that of our system is also constant to be 7.238 s. Since the calculated relevance score of KNN-SE is plaintext, its time consumption is smaller than ours. Our system has better efficiency in *KeyGen*, *BuildIndex*, *Trapdoor* algorithms.

6 SECURITY ANALYSIS

In this section, we prove the security of the protocols under the attack model (defined in Section 2.2) and security model (defined in Section 2.3). Then, we analyze the security of the proposed novel MRSE system.

6.1 Protocols Security Proof

Theorem 1. *The **KET** protocol proposed in Section 3.3 is secure to test the equality on two keyword ciphertext in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Please refer Supplemental Materials C, available online for the security proof.

Theorem 2. *The **KET** protocol is secure against the adversary \mathcal{A}^* defined in the attack model.*

Please refer Supplemental Materials C, available online for the security proof.

Theorem 3. *The **MKS** protocol proposed in Section 4.5 is secure to calculate the relevance score on encrypted index and query in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Proof. **MKS** protocol calls **KET** and **SMD** as subprotocols and all the transmitted data are encrypted using PCTD encryption. Since **KET** and **SMD** protocols are proved secure in Theorem 1 and [40], the **MKS** protocol is also secure in the presence of attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$. \square

Theorem 4. *The **MKS** protocol is secure against the adversary \mathcal{A}^* defined in the attack model.*

Proof. **MKS** protocol calls **KET** and **SMD** as subprotocols and all the transmitted data are encrypted using PCTD encryption. The **KET** protocol is proved secure against the adversary \mathcal{A}^* (defined in the attack model) in Theorem 2, and the **SMD** protocol is proved secure in [40]. Thus, the **MKS** protocol is also secure against the adversary \mathcal{A}^* defined in the attack model. \square

Theorem 5. *The **MAX** protocol proposed in Section 4.5 is secure to calculate the maximum encrypted data (with highest relevance score) on two encrypted ciphertext in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Please refer Supplemental Materials C, available online for the security proof.

Theorem 6. *The **MAX** protocol is secure against the adversary \mathcal{A}^* defined in the attack model.*

Please refer Supplemental Materials C, available online for the security proof.

Theorem 7. *The **MAX_n** protocol proposed in Section 4.5 is secure to calculate the maximum encrypted data (with highest relevance score) on n encrypted ciphertext in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Proof. **MAX_n** protocol calls **MAX** as subprotocol and all data are encrypted using PCTD encryption. Since **MAX** protocol is proved secure in Theorem 5, the **MAX_n** protocol is also secure in the presence of attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$. \square

Theorem 8. *The **MAX_n** protocol is secure against the adversary \mathcal{A}^* defined in the attack model.*

Proof. **MAX_n** protocol calls **MAX** as subprotocol and all data are encrypted using PCTD encryption. Since **MAX** protocol is proved secure against the adversary \mathcal{A}^* (defined in the attack model) in Theorem 6, the **MAX_n** protocol is also secure against the adversary \mathcal{A}^* defined in the attack model. \square

Theorem 9. *The **TOP-k** protocol proposed in Section 4.5 is secure to calculate the top-k encrypted data (with highest relevance scores) on encrypted ciphertext in the presence of semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Please refer Supplemental Materials C, available online for the security proof.

Theorem 10. *The **TOP-k** protocol is secure against the adversary \mathcal{A}^* defined in the attack model.*

Please refer Supplemental Materials C, available online for the security proof.

6.2 System Security Analysis

In this section, we analyze the privacy of each algorithms in novel MRSE system.

- Key Generation: The security of the user's private key is guaranteed by discrete logarithm problem.

The security of the secret keys of CP, CSP and KGC is ensured by the IND-CPA security of PCTD.

- User Authorization and Revocation: Since the users and KGC's private keys are kept secret and the signature system is cryptographically strong unforgeable, the security of authorization and revocation certificates is guaranteed.
- Encryption: Since the keywords, file identity and symmetric key are encrypted using PCTD encryption algorithm, the encrypted index is secure. The privacy of file is ensured by the semantic security of the symmetric encryption algorithm.
- Query: The security analysis of Query algorithm is similar to Encryption algorithm.
- Search: In the search phase, the privacy-preserving relevance scores are calculated using **MKS** protocol. Then, the top- k encrypted files are calculated using **TOP- k** protocol. Since these two protocols are proved secure and all transmitted data are ciphertext, the privacy of search algorithm is ensured.

Next, we utilize the attack model in Section 2.2 to analyze that our system is secure against adversary \mathcal{A}^* .

- 1) If \mathcal{A}^* eavesdrops all the contents that are transmitted between the challenge data user (including data owner) and CP. The encrypted files and indexes and the queried results can be gotten by \mathcal{A}^* . The intermediate calculated ciphertext that are transmitted between CP and CSP can also be eavesdropped by \mathcal{A}^* . Since these contents are all protected by PCTD encryption during the transmission process, \mathcal{A}^* is not capable to derive the plaintext without the secret keys of data user, CP or CSP.
- 2-3) Assume \mathcal{A}^* compromise CP or CSP and get the partial strong private key λ_1 or λ_2 . However, \mathcal{A}^* can not simultaneously compromise CP and CSP. \mathcal{A}^* could not get the strong secret key λ since it is randomly split into two parts using "strong secret key splitting" algorithm of PCTD. Even though \mathcal{A}^* could compromise CSP and get the plaintext of intermediate result in these protocols, he is not able to get any useful information since the "blinding" method [54] is utilized in these protocols to conceal the plaintext: A random number is added to or multiplied with the message before they are sent to CSP.
- 4) If the adversary \mathcal{A}^* gets the secret keys of data owners or data users (except the challenge user's secret key), he could not decrypt the challenge user's ciphertext. The reason is that the private keys of different users are irrelevant.

In Theorems 1-10, the protocols **KET**, **MKS**, **MAX**, **MAX_n**, **TOP- k** are proved secure against the adversary \mathcal{A}^* defined in the attack model and the semi-honest (non-colluding) attackers $\mathcal{A} = (\mathcal{A}_{D_1}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ defined in the security model. Thus, this system is also secure against these adversaries.

7 CONCLUSION

Multi-keyword rank searchable encryption is a useful technique which allows data users to search over encrypted data in the cloud. Many MRSE systems have been proposed in the literature and most of them are constructed based on

the KNN-SE algorithms. In this paper, we first pointed out several serious shortcomings of KNN-SE which limit the practical applications of the existing MRSE systems. We then proposed a new MRSE system which overcomes all the defects of the KNN-SE based MRSE systems. Our new system does not require a predefined keyword set at the system setup phase and support keyword search in arbitrary languages. The system allows flexible search authorization and time-controlled revocation. In addition, the relevance scores computed by the cloud server are in ciphertext form and the server is not able to tell which documents are the top- k results. We proved the security of the system and conducted extensive computer simulations to demonstrate its efficiency.

ACKNOWLEDGMENTS

The authors thank the associate editor and reviewers for their constructive and generous feedback. This work is supported by National Natural Science Foundation of China (61402112, 61702105, 61672159); Singapore National Research Foundation (NRF2014NCR-NCR001-012); AXA Research Fund; Technology Innovation Platform Project of Fujian Province (2014H2005).

REFERENCES

- [1] M. Armbrust, et al., "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [2] J. W. Rittinghouse and J. F. Ransome, *Cloud Computing: Implementation, Management, and Security*. Boca Raton, FL, USA: CRC Press, 2016.
- [3] N. H. Hussein and A. Khalid, "A survey of cloud computing security challenges and solutions," *Int. J. Comput. Sci. Inform. Security*, vol. 14, no. 1, 2016, Art. no. 52.
- [4] M. Abdalla, et al., "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Annual International Cryptology Conference*. Berlin Heidelberg, Germany: Springer, 2005, pp. 205–222.
- [5] K. Tomida, H. Doi, M. Mohri, and Y. Shiraishi, "Ciphertext divided anonymous HIBE and its transformation to identity-based encryption with keyword search," *J. Inf. Process.*, vol. 23, no. 5, pp. 562–569, 2015.
- [6] Y. Yang and M. Ma, "Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds," *IEEE Trans. Inform. Forensics Security*, vol. 11, no. 4, pp. 746–759, Apr. 2016.
- [7] Y. Wu, X. Lu, J. Su, and P. Chen, "An efficient searchable encryption against keyword guessing attacks for sharable electronic medical records in cloud-based System," *J. Med. Syst.*, vol. 40, no. 12, 2016, Art. no. 258.
- [8] Y. Yang, "Attribute-based data retrieval with semantic keyword search for e-health cloud," *J. Cloud Comput.*, vol. 4, no. 1, 2015, Art. no. 1.
- [9] W. Wen, R. Lu, J. Lei, H. Li, X. Liang, and X. Shen, "SESA: An efficient searchable encryption scheme for auction in emerging smart grid marketing," *Security Commun. Netw.*, vol. 7, no. 1, pp. 234–244, 2014.
- [10] Y. Yang, X. Zheng, and C. Tang, "Lightweight distributed secure data management system for health internet of things," *J. Netw. Comput. Appl.*, vol. 89, pp. 26–37, 2017.
- [11] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inform. Forensics Security*, vol. 11, no. 4, pp. 789–798, 2016.
- [12] Y. Yang and M. Ma, "Semantic searchable encryption scheme based on lattice in quantum-era," *J. Inform. Sci. Eng.*, vol. 32, no. 2, pp. 425–438, 2016.
- [13] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *International Conference on Applied Cryptography and Network Security*. Berlin Heidelberg, Germany: Springer, 2004, pp. 31–45.

- [14] J. W. Byun, D. H. Lee, and J. Lim, "Efficient conjunctive keyword search on encrypted data storage system," in *European Public Key Infrastructure Workshop*. Berlin Heidelberg, Germany: Springer, 2006, pp. 184–196.
- [15] Y. Yang, M. Ma, and B. Lin, "Proxy re-encryption conjunctive keyword search against keyword guessing attack," in *Proc. Comput. Commun. IT Appl. Conf.*, 2013, pp. 125–130.
- [16] H. T. Poon and A. Miri, "An efficient conjunctive keyword and phase search scheme for encrypted cloud storage systems," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, 2015, pp. 508–515.
- [17] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.
- [18] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 23rd ACM SIGSAC Conf. Comput. Commun. Security*, 2006, pp. 79–88.
- [19] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Proc. Annu. Int. Cryptology Conf.*, 2013, pp. 353–373.
- [20] J. K. Liu, M. H. Au, W. Susilo, K. Liang, R. Lu, and B. Srinivasan, "Secure sharing and searching for real-time video data in mobile cloud," *IEEE Netw.*, vol. 29, no. 2, pp. 46–50, Mar./Apr. 2015.
- [21] X. Yang, T. T. Lee, J. K. Liu, and X. Huang, "Trust enhancement over range search for encrypted data," in *Proc. IEEE Int. Conf. Trust Security Privacy Comput. Commun.*, 2016, pp. 66–73.
- [22] C. Zuo, J. Macindoe, S. Yang, R. Steinfeld, and J. K. Liu, "Trusted boolean search on cloud using searchable symmetric encryption," in *IEEE Int. Conf. Trust Security Privacy Comput. Commun.*, 2016, pp. 113–120.
- [23] S. F. Sun, J. K. Liu, A. Sakzad, R. Steinfeld, and T. H. Yuen, "An efficient non-interactive multi-client searchable encryption with support for boolean queries," in *European Symposium on Research in Computer Security, LNCS 9878*. Berlin, Germany: Springer, 2016, pp. 154–172.
- [24] S. K. Kermanshahi, J. K. Liu, and R. Steinfeld, "Multi-user cloud-based secure keyword search," in *Australasian Conference on Information Security and Privacy, LNCS 10342*. Berlin, Germany: Springer, 2017, pp. 227–247.
- [25] D. Boneh, G. Di Crescenzo, G. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, vol. 3027, pp. 506–522.
- [26] K. Liang, C. Su, J. Chen, and J. K. Liu, "Efficient multi-function data sharing and searching mechanism for cloud-based encrypted data," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, 2016, pp. 83–94.
- [27] K. Liang, X. Huang, F. Guo, and J. K. Liu, "Privacy-preserving and regular language search over encrypted cloud data," *IEEE Trans. Inform. Forensics Security*, vol. 11, no. 10, pp. 2365–2376, Oct. 2016.
- [28] N. Cao, C. Wang, M. Li, K. Ren, and L. Wenjing, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE Conf. Comput. Commun.*, 2011, pp. 829–837, doi: [10.1109/INFCOM.2011.5935306](https://doi.org/10.1109/INFCOM.2011.5935306).
- [29] J. Yu, P. Lu, Y. Zhu, G. Xue, and L. Minglu, "Toward secure multi-keyword top-k retrieval over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 10, no. 4, pp. 239–250, Jul./Aug. 2013.
- [30] Z. Fu, X. Sun, N. Linge, and L. Zhou, "Achieving effective cloud search services: Multi-keyword ranked search over encrypted cloud data supporting synonym query," *IEEE Trans. Consum. Electron.*, vol. 60, no. 1, pp. 164–172, Feb. 2014.
- [31] Z. Fu, J. Shu, X. Sun, and N. Linge, "Smart cloud search services: Verifiable keyword-based semantic search over encrypted cloud data," *IEEE Trans. Consum. Electron.*, vol. 60, no. 4, pp. 762–770, 2014.
- [32] W. Sun, et al., "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 3025–3035, Nov. 2014.
- [33] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerging Topics Comput.*, vol. 3, no. 1, pp. 127–138, Mar. 2015.
- [34] H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: When QOE meets QOP," *IEEE Wireless Commun.*, vol. 22, no. 4, pp. 74–80, Aug. 2015.
- [35] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 3, pp. 312–325, May/June 2016.
- [36] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.
- [37] C. Chen, et al., "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.
- [38] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inform. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.
- [39] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Berlin Heidelberg, Germany: Springer, 1999, pp. 223–238.
- [40] X. Liu, R. H. Deng, K. K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkits with multiple keys," *IEEE Trans. Inform. Forensics Security*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.
- [41] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Berlin Heidelberg, Germany: Springer, 2003, pp. 37–54.
- [42] X. Liu, R. H. Deng, W. Ding, R. Lu, and B. Qin, "Privacy-preserving outsourced calculation on floating point numbers," *IEEE Trans. Inform. Forensics Security*, vol. 11, no. 11, pp. 2513–2527, Nov. 2016.
- [43] X. Liu, R. Choo, R. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Dependable Secure Comput.*, Mar. 2016, doi: [10.1109/TDSC.2016.2536601](https://doi.org/10.1109/TDSC.2016.2536601).
- [44] Q. Do, B. Martini and K. K. R. Choo, "A forensically sound adversary model for mobile devices," *PLoS ONE*, vol. 10, no. 9, 2015, Art. no. e0138449.
- [45] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multiparty computation," in *Proc. Int. Assoc. Cryptologic Res. Cryptology ePrint Archive*, 2011, Art. no. 272.
- [46] X. Liu, B. Qin, R. H. Deng, and Y. Li, "An efficient privacy-preserving outsourced computation over public data," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 756–770, Sep./Oct. 2017, doi: [10.1109/TSC.2015.2511008](https://doi.org/10.1109/TSC.2015.2511008).
- [47] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 2112–2120.
- [48] Unicode Consortium. *The Unicode Standard, Version 2.0*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 1997.
- [49] R. Ostrovsky and W. E. Skeith III, "A survey of single-database private information retrieval: Techniques and applications," in *International Workshop on Public Key Cryptography*. Berlin Heidelberg, Germany: Springer, Apr. 2007, pp. 393–411.
- [50] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III, "Public key encryption that allows PIR queries," in *Proc. Annu. Int. Cryptology Conf.*, 19 Aug. 2007, pp. 50–67.
- [51] E. Stefanov, et al., "Path ORAM: An extremely simple oblivious RAM protocol," in *Proc. 23rd ACM SIGSAC Conf. Comput. Commun. Security*, 4 Nov. 2013, pp. 299–310.
- [52] D. Cash, A. Kuppuc, and D. Wichs, "Dynamic proofs of retrievability via oblivious ram," *J. Cryptology.*, vol. 30, no. 1, pp. 22–57, Jan. 2017.
- [53] E. Barker, E. Barker, W. Burr, W. Polk, and M. Smid, "NIST special publication 800-57," *NIST Special Publication*, vol. 800, no. 57, pp. 1–142, 2007.
- [54] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Trans. Inform. Forensics Security*, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.



Yang Yang (M'16) received the BSc degree from Xidian University, Xi'an, China, and the PhD degree from Xidian University, China, in 2006 and 2012. She is an associate professor in the College of Mathematics and Computer Science, Fuzhou University. She is also a research fellow (postdoctor) under supervisor Robert H. Deng in the School of Information System, Singapore Management University. She has published more than 40 research articles include the *IEEE Transactions on Information Forensics and Security*,

the *IEEE Transactions on Dependable and Secure Computing* and the *IEEE Transactions on Services Computing*. Her research interests include the area of cloud, IoT and big data security, and privacy protection. He is a member of the IEEE.



Ximeng Liu (S'13-M'16) received the BSc degree in electronic engineering from Xidian University, Xi'an, China, and the PhD degree in cryptography from Xidian University, China, in 2010 and 2015. Now, he is a research fellow in the School of Information System, Singapore Management University, Singapore, and Qishan Scholar in the College of Mathematics and Computer Science, Fuzhou University. He has published more than 80 research articles include the *IEEE Transactions on Information Forensics and*

Security, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Services Computing* and the *IEEE Transactions on Cloud Computing*. His research interests include cloud security, applied cryptography, and big data security. He is a member of the IEEE.



Robert H. Deng (F'16) is an AXA chair professor of cybersecurity in the School of Information Systems, Singapore Management University. His research interests include data security and privacy, network and system security. He has served/is serving on the editorial boards of many international journals in security, such as the *IEEE Transactions on Information Forensics and Security*, the *IEEE Transactions on Dependable and Secure Computing*, the *International Journal of Information Security*, and the *IEEE Security and Privacy Magazine*. He is fellow of the IEEE.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Pattern-Growth Based Mining Mouse-Interaction Behavior for an Active User Authentication System

Chao Shen¹, Yufei Chen¹, Xiaohong Guan, *Fellow, IEEE*, and Roy A. Maxion, *Fellow, IEEE*

Abstract—Analyzing mouse-interaction behaviors for implicitly identifying computer users has received growing interest from security and biometric researchers. This study presents a simple but efficient active user authentication system by modeling mouse-interaction behavior, which is accurate and competent for future deployments. A pattern-growth-based mining method is proposed to extract frequent behavior segments, in obtaining a stable and discriminative representation of mouse-interaction behavior. Then procedural features are extracted to provide an accurate and fine-grained characterization of the behavior segments. A SVM-based decision procedure using one-class learning techniques is applied to the feature space for performing authentication. Analyses are conducted using data from around 1,526,400 mouse operations of 159 participants, and the authentication performance is evaluated across various application scenarios and tasks. Our experimental results show that characteristics from frequent behavior segments are more stable and discriminative than those from holistic behavior, and the system achieves a practically useful level of performance with FAR of 0.09 percent and FRR of 1 percent. Additional experiments on usability to sample length, reliability to application task, scalability to user size, robustness to mimic attack, and response to behavior change are provided to further explore the applicability. We also compare the proposed approach with the state-of-the-art approaches for the collected data.

Index Terms—Anomaly detection, mouse-interaction behavior, one-class learning, pattern growth, active authentication

1 INTRODUCTION

EXTERNAL attackers and insiders masquerading as legitimate users have always been a serious problem in cyber-security settings. The threats from these attackers, who abuse the privileges for malicious purposes, have overtaken malwares as the most reported security incident according to recent reports from US National Threat Assessment Center [15] and Australian Cyber Security Centre [24], [26]. Thanks to leaked or fabricated identity credentials, impostors can access computer systems easily, and abuse available privileges to masqueraders. The most common approach to address this problem is user authentication mechanism. Unfortunately, most existing computer and network systems authenticate a user only at the login moment. Since the authentication occurs only once, attackers may take control of user sessions. In contrast, active (re)authentication, which is done throughout the session, can prevent such an attack. To achieve a timely and accurate response without the user involvement, the authentication should be transparent to users.

Of various potential solutions to this problem, a promising technique is the use of mouse-interaction behavior

- C. Shen, Y. Chen, and X. Guan are with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an 710049, China. E-mail: {cshen, yfchen, xhguan}@sei.xjtu.edu.cn.
- R. Maxion is with Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213. E-mail: maxion@cs.cmu.edu.

Manuscript received 29 Dec. 2016; revised 22 Sept. 2017; accepted 9 Oct. 2017. Date of publication 8 Nov. 2017; date of current version 18 Mar. 2020.

(Corresponding author: Chao Shen.)

Digital Object Identifier no. 10.1109/TDSC.2017.2771295

[4], [36], which offers the ability to ascertain a user's mouse behavioral characteristics to verify her/his identity in a transparent manner. As a behavioral biometric, this behavior has been strongly driven by the need for nonintrusive authentication for detection and monitoring applications. It has recently experienced growing interest, e.g., the Active Authentication and Monitoring Program sponsored by DARPA [3]. Additionally, it requires no specialized hardware to capture biometric data, and is less likely to be obscured than other biometrics [4]. Moreover, the detection process can be integrated seamlessly into users' computer usage, and thus provide a non-intrusive solution for identity monitoring or active authentication after initial authentication by passwords or other credentials.

Although previous work has reported some promising results, mouse-behavior based authentication is still a newly-emerged technique and has not reached an acceptable performance. One of the major reasons is behavioral variability, in contrast with other physiological biometric characteristics, such as face or fingerprint patterns. Behavioral variability can occur between two consecutive samplings, even if a user provides the samples striving to maintain a uniform way of mouse operations. This variability often comes from intrinsic human factors or external environmental variables, such as changes in software environments or interaction modes; it sometimes relates to variations in biological or emotional status of the operators [7], [10]. This may partially explain why accuracies of using this biometric for authentication are

reported with mixed results, with the equal-error rate (ERR) ranging from 24.3 to 1.3 percent [1], [2], [5], [6], [7], [8], [21], [23], [32], [34], [34], [37], [38], [39]. Although the behavioral variability is an important issue in mouse-behavior analysis, previous approaches mainly focus on the discriminative power of this behavior. Moreover, most previous work has based their designs on a straightforward idea that mouse operations can be utilized to identify users. Yet, the biometric properties of mouse behavior across different application tasks and scenarios have not been comprehensively evaluated. It is also notable that most research has used mouse data from both impostors and legitimate users to train authentication model, which may limit its applicability.

1.1 Overview of Our Approach

In this paper, we present an active and transparent authentication approach by analyzing a user's mouse-interaction behavior across various operational and application scenarios. We exploit a mining method of extracting repeatable behavior segments of mouse behavior to obtain stable behavioral features, and develop a decision procedure using one-class classification algorithms to perform continuous user authentication. The main purpose and contributions of this work are five folds.

First, we model behavioral biometrics using mouse-interaction behavior for active user authentication across various application conditions, and investigate distinctiveness and permanence properties of the behavior. We develop an efficient authentication system, which is easy to implement and can verify a user in high accuracy and a short time, with minor system overhead.

Second, we propose a pattern-growth-based mining method to extract frequent behavior segments in obtaining stable and discriminative mouse characteristics, which generates an accurate and stable representation of mouse behavior. This can greatly reduce behavioral variability, and lead to a performance improvement of authentication accuracy and time.

Third, we propose a set of mouse-behavior features by characterizing various operational properties, and make a systematic exploration on their discriminability and stability. We then employ one-class learning methods to build an authentication model, so that the model can be trained solely on samples from the legitimate user. We also ensure the diversity in a set of detectors to compare authentication performance, for discovering promising strategies for modeling mouse behavior.

Fourth, we examine the proposed approach in terms of reliability to operational scenario, usability to sample length, sensitivity to application task, robustness to mimic attack, and response to behavior change, to further examine its applicability and generalization capability. We also provide both qualitative and quantitative comparisons with the state-of-the-art approaches.

Fifth, we develop and implement a simple but efficient active authentication system to make it respond to users' presence, and to protect interactive usage sessions. The performance shows mouse characteristics extracted from frequent behavior segments are much more stable than those from holistic behavior, and achieve a practically useful level. The results suggest that mouse-interaction behavior

suffice to be a significant enhancement for a traditional authentication system, which demonstrates a shift from passive to active security control.

This paper is organized as: Section 2 reviews related research. Section 3 introduces mouse-interaction behavior. Sections 4, 5, and 6 describe behavior pattern mining method, and develop a decision procedure. Section 7 presents evaluation methodology. Section VIII presents results. Section 9 offers discussions, and Section 10 concludes.

2 BACKGROUND AND RELATED WORK

2.1 Review of Mouse-Behavior Analysis

Mouse-behavior analysis, as a behavioral biometric [4] using mouse-interaction data, can provide user authentication in an accessible and convenient manner.

Since Everitt and McOwan [21] first investigated in 2003 whether users could be distinguished by their mouse operating styles, different techniques and usages of mouse behavior have been proposed. Among the investigations of mouse-behavior analysis for authentication, there are really two tasks of interest [36]: static authentication and active (re) authentication. The first application usually checks a user only once, typically at the login time [2], [21], [34], [37], [38]. These approaches commonly require the user to perform a pre-designed sequence of mouse actions, which are analyzed to form feature vectors for being compared with a legitimate user's profile to verify the identity. The second application passively acquires a user's mouse data during computer usage sessions, and analyzes behavior features to implicitly verify continued presence of the user throughout the session [6], [8], [20], [32], [33], [39]. These efforts reveal mouse behavior has a rich potential for authentication, not only at the login time, but throughout user sessions. Yampolskiy et al. [4] and Jorgensen et al. [16] provide good reviews of this field.

2.2 Mouse Behavior for Active Authentication

An important strength of mouse behavior is to constantly monitor users' sessional usage of computer systems, and to enable nonintrusive authentication. This study focuses on active (re)authentication for mouse behavior.

Pusara and Brodley [1] were among the earliest researchers to explore the possibility of using mouse behavior for active (re)authentication. They characterized mouse clicks and movements in a window, and set up a personalized model using decision tree. Based on data collected in a free environment, a false-acceptance rate (FAR) of 1.75 percent and false-rejection rate (FRR) of 0.43 percent were reported, with time ranging from 1 to 15 minutes. But the study is inconclusive with only 11 users, prompting the authors to conclude mouse behavior is insufficient for authentication. Our study relies on an improved methodology and more users, leading us to reverse their hypothesis.

Ahmed and Traore [6] considered mouse behavior for intrusion detection and (re)authentication. They aggregated low-level mouse events as higher-level actions, and extracted a 39-dimensional feature vector from a block of mouse actions. They used a neural network for model training and classification. Based on the data from 22 subjects, they achieved an EER of 2.46 percent, with about 2000 mouse actions required for authentication. Conversely, our

TABLE 1
Mouse Event

Event Name	Description
Mouse Down	A user presses left/right/middle mouse button.
Mouse Up	A user releases left/right/middle mouse button.
Mouse Wheel	A user moves mouse wheel.
Mouse Move	A user moves the mouse.

study aims to find stable mouse-behavior patterns which would significantly reduce the required mouse actions for authentication.

Gamboa and Fred [5], [11] proposed an active (re)authentication approach based on mouse movements. Each movement was characterized by a 63-dimensional feature vector including spatial parameters and temporal parameters. They applied greedy feature selection, and made authentication decision on average classification of a movement sequence using a statistical model. After performing experiments on data from 50 users under a free environment, they found that sequences of 1, 50, and 100 movements yielded an equal error rate of 48.9, 2 and 0.7 percent respectively. However, the test data were also used for feature selection, which may lead to an overly optimistic performance.

Later on, Mondal and Bours [33] and Rahman et al. [31] built lightweight continuous verification systems via users' mouse activities. Mondal and Bours [33] characterized every mouse action, and established an authentication model using six machine learning algorithms. Based on 45 users' data, a 0 percent FAR was obtained with the requirement of 94 mouse actions for authentication. Rahman et al. [31] compared mouse activities against a simple statistical profile, and tested the approach using data from 45 users. They achieved an average EER of 13.42 percent.

Recently, Zheng et al. [25], [32] presented an active (re) authentication system based on fine-grained (point-by-point) angle-based metrics of mouse movements. They utilized Support Vector Machines for quick and accurate classification. Based on data collected in controllable environments and in the field, they reported an EER of 1.3 percent with about 20 mouse clicks. They also discussed the effects of environmental factors. They also pointed out that the reliability of this behavior across different scenarios needs to be addressed for putting it into more practice.

As compared with other biometric features, mouse behavior may be with a lack of in-depth analysis. This study, differing from existing work: (1) aims to provide in-depth analysis of mouse behavior for active authentication in terms of discriminability, stability, and applicability, and to address mouse-behavior variability; (2) evaluates distinctiveness and permanence properties of mouse behavior across different application scenarios and tasks, instead of basing on the assumption that mouse behavior qualifies good biometrics; (3) employs one-class learning methods to build an authentication model, which is based solely on training data from legitimate user; (4) explores the effectiveness of this technique across various application scenarios and tasks; (5) examines a set of detectors to compare the performance and whether an observed effect is specific to one

TABLE 2
Mouse Action (with the number of action types in parentheses.)

Mouse Actions	Description
Single click	Mouse down event followed by mouse up event of left/right/middle button (3)
Double click	A set of mouse down, up, down and up event of left/right/middle button (3)
Common movement	General mouse movement involving no clicks (1)
Point and click movement	A mouse movement followed by single or double clicks at the end (1)
Drag and drop movement	An action starting with mouse down, followed by a movement and ending with mouse up (1)
Silence	Standstill of mouse cursor (1)

detector; (6) implements an authentication system integrated into Windows operating system.

3 MOUSE-BEHAVIOR ANALYSIS

3.1 Mouse Event

Mouse behavior is commonly described as a stream of mouse events received from mouse input devices. The first step to understand mouse behavior is to recognize mouse events from collected raw data stream. In general, the collected data are a list of events such as mouse movement, mouse button down, mouse button up, and so on. Table 1 shows relevant mouse events in mouse behavior.

3.2 Mouse Operation

While mouse interactions can be interpreted in an event-driven way, such raw events do not provide meaningful information for analyzing behavior. Thus it is necessary to translate these events into meaningful actions. Table 2 lists common mouse actions and corresponding event-level interpretation, with the number of the action types in parentheses. Each action corresponds to a set of consecutive mouse events. In this study, we encoded mouse actions into mouse operations, along with the application information. Each mouse operation is represented as a tuple of multi-attributes and timestamp, which is in the form of (action-type, application-type, screen-area, window-position, timestamp). Table 3 summarizes the 5 attributes for a mouse operation, in which the third column lists the encoding number in sequence for this evaluation.

3.3 Mouse-Behavior Pattern

Through a preliminary analysis of mouse operations from 20 users in their daily usage of computers, we discovered that some behavior segments, consisting of a series of consecutive operations, would recur frequently in users' routine mouse usage. In this paper, we refer to the recurring behavior segment as a *behavior pattern*. This study divides the behavior patterns into two categories: micro-habitual patterns and task-intended patterns. Micro-habitual patterns characterize subconscious and habitual constituents of mouse activities during interactions, such as some habitual mouse operations. For instance, most subjects are accustomed to repeatedly refresh computer screen, which means the subject would click right mouse button on an empty

TABLE 3
Mouse Operation

Attribute	Descriptions	Encoding values (in sequence)
Action type	10 mouse actions in Table 2.	0-9
Application type	Application in which the action occurs, including Internet surfing, word processing, online chatting, and game playing, desktop operating, and others.	0-5
Screen area	9 evenly divided regions in the area of mouse cursor on screen.	0-8
Window position	The position of window, including client area, close area, maximum area, minimum area, menu, and title bar.	0-5
Timestamp	The timestamp when the action occurs.	-

screen area, and then select “Refresh” from the contextual menu, which corresponds to a series of mouse operations: right single click \rightarrow mouse movement \rightarrow left single click. Task-intended patterns characterize operational agility and habits of individual mouse activities under certain applications, such as regularly utilizing certain functions. Concretely, if the subject wants to create a new document in a word processor, he/she would first click left mouse button on the menu of the word processor, next select “New” from the menu, and then select “blank document” from the new contextual panel and double click it, which corresponds to a group of mouse operations: left single click \rightarrow movement \rightarrow left single click \rightarrow movement \rightarrow left double click.

Moreover, upon a closer examination of the behavior patterns, we found relevant metrics extracted from behavior patterns appeared much more stable than those from holistic behavior. We conjectured that the improved stability may be due to recurrent behavior segments providing more stable and habitual information about mouse behavior. Therefore, we made an assumption that frequent behavior segments in mouse behavior can provide more stable and discriminative measurements.

4 MOUSE-BEHAVIOR PATTERN MINING

4.1 Problem of Mouse-Behavior Pattern Mining

The goal is to extract frequent mouse behavior patterns from holistic behavior, thus we first define the problem of mouse behavior pattern mining. Let x be a mouse operation in the form of (action type, application-type, screen-area, window-position, timestamp) (e.g., $(1, 3, 4, 0, t)$, given the definition in Table 3), then an operation set $os = \langle x_1, x_2, \dots, x_m \rangle$, $m \geq 1$ is denoted as multiple consecutive mouse operations (e.g., $\langle (1, 3, 4, 0, t_0) \rangle$ or $\langle (1, 3, 4, 0, t_1), (2, 1, 4, 0, t_2), (3, 1, 4, 0, t_3) \rangle$). An operation sequence is considered as a time-ordered list of operations set by User ID and timestamp, which is denoted as $s = \{os_1, os_2, \dots, os_l\}$, $l \geq 1$ (e.g., $\{(1, 3, 4, 0), (1, 3, 4, 0), \dots, (2, 1, 4, 0), (3, 1, 4, 0), \dots, (2, 1, 2, 1)\}$, here we omit the timestamp since the elements in the sequence are time-ordered). A mouse operation

TABLE 4
An Instance in Mouse Operation Database

User ID	Sequence ID	Sequence
1	1	$\{(1, 3, 4, 0), (1, 3, 4, 0), (2, 1, 4, 0), (3, 1, 4, 0), \dots, (2, 1, 2, 1)\}$
1	2	$\{(2, 1, 4, 0), (1, 1, 5, 1), \dots, (3, 1, 4, 0)\}$
1	3	$\{(2, 3, 4, 0), (1, 1, 3, 0), \dots, (1, 2, 2, 5)\}$
...

sequence database S is a set of triples $\langle ID, sid, s \rangle$, where ID is the user ID, sid is the sequence ID, and s is the corresponding operation sequence. A triple $\langle ID, sid, s \rangle$ is said to contain a sequence α , if α is a subsequence of s . The support of a sequence α in a database S is the number of tuples in the databases containing α . It can be denoted as $support_S(\alpha)$ if the sequence database is clear from the context. Given a positive integer ξ as the support threshold, a sequence α is called a sequential pattern in sequence database S if the sequence is contained by at least ξ tuples in the database, i.e., $support_S(\alpha) \geq \xi$. A sequential pattern with length l is called an l -pattern. In this study, we recorded mouse operations in a sequence database, and each sequence is composed by some consecutive operations between two adjacent silence actions.

An instance from mouse operation sequence database is shown in Table 4, and the third column corresponds to the action type defined in Table 3. Taking the first entry in Table 4 as an example, a sequence $\{(1, 3, 4, 0), (1, 3, 4, 0), (2, 1, 4, 0), (3, 1, 4, 0), \dots, (2, 1, 2, 1)\}$ has several elements: $\langle (1, 3, 4, 0) \rangle$, $\langle (1, 3, 4, 0), (2, 1, 4, 0) \rangle$, $\langle (1, 3, 4, 0), (3, 1, 4, 0) \rangle$, \dots , where mouse operation $(1, 3, 4, 0)$ appears more than once respectively in different elements. Mouse operation $(1, 3, 4, 0)$ happens twice in this sequence, so it contributes 2 to the length of the sequence.

Problem Statement. Given a mouse sequence database S and a minimum support threshold ξ , the problem of mouse behavior pattern mining is to find the complete set of frequent behavior patterns in the database.

4.2 Mouse Behavior Pattern Mining Method

While most researchers mainly used holistic behavior for authentication, we developed a pattern-growth based [19] sequential mining method, as the basic engine for mining mouse behavior patterns from observed mouse operations. Its major idea is to examine prefix subsequences and project their corresponding suffix subsequences into projected databases. In each projected database of mouse behavior, sequential patterns grow by exploring only local frequent patterns. It mines the complete set of sequential patterns and substantially reduces efforts of candidate subsequence generation. Here we first introduced conceptions of prefix, suffix and projected database.

Prefix. Given the sequence $\alpha = e_1 e_2 \dots e_n$ (where each e_i corresponds to a frequent element in S), sequence $\beta = e'_1 e'_2 \dots e'_m$ ($m < n$) is called a prefix of α if and only if (1) $e'_i = e_i$ for $i \leq m - 1$; (2) $e'_m e_m i$ and (3) all the items in $e_m - e'_m$ are alphanumerically after those in e'_m .

Suffix. Given a sequence $\alpha = e_1 e_2 \dots e_n$ (where each e_i corresponds to a frequent element in S). Let $\beta = e_1 e_2 \dots e_{m-1} e'_m$ ($m < n$) be the prefix of α . Sequence $\gamma = e''_m e_{m+1} \dots e_n$ ($m < n$) is called the suffix of α with regards

Input: A mouse operation sequence database S , and a minimum support threshold ξ .

Output: Complete set of frequent mouse behavior patterns P .

Method: *MouseBehaviorPatternMining()*

Call *MouseBehaviorPatternMining* (\diamond, S, ξ)

Begin

```

(1) Let  $\alpha$  be a sequential mouse operation pattern;
(2) if  $\alpha \neq \diamond$ 
(3)   Let  $SI_\alpha$  be the  $\alpha$ -projected database;
(4) else
(5)    $SI_\alpha = S$ ;
(6) end
(7) Let  $l$  be the length of  $\alpha$ ,  $B = \{\}$ ;
(8) for each mouse operation in  $SI_\alpha$  do begin
(9)   Find the set of frequent item  $\{b_i\}$ ;
(10)  For each  $b_i$  satisfying the product of frequency of  $b_i$  in
       $SI_\alpha$  and  $(l+1)$  is larger than  $\xi$  then begin
(11)    if  $b_i$  can be assembled to the last element of
(12)      form a new set  $B = B \cup \{b_i\}$ ;
(13)    end
(14)  end
(15) end
(16) for each frequent operation  $b$  in  $B$  do begin
(17)    $\alpha' = \alpha \cup \{b\}$ ;
(18)   form a new  $\alpha'$ -projected database  $SI_{\alpha'}$ ;
(19)   Call MouseBehaviorPatternMining ( $\alpha', SI_{\alpha'}, \xi$ );
(20) end
end

```

Fig. 1. Mouse behavior pattern mining algorithm.

to prefix β , denoted as $\gamma = \alpha/\beta$, where $e_m'' = (e_m - e_m')$. Note that if β is not a subsequence of α , the suffix of α with respect to β is empty.

Projected Database. Let α be a sequential pattern in sequence database S . The α -projected database, denoted as SI_α , is the collection of suffixes of sequences in S w.r.t. prefix α .

The behavior patterns are incrementally extracted from the mouse operation sequence database using the algorithm in Fig. 1. This procedure first scans the mouse operation database S once to find length-1 mouse behavior patterns, then divides the search space into *prefixes*, and finally constructs the corresponding set of *projected databases*, each of which would be mined recursively, to find subsets of sequential mouse behavior patterns.

4.3 Reference Patterns Generation and Matching

When applying the proposed pattern mining algorithm to extract behavior patterns from holistic mouse behavior, the first aim is to create “baseline” normal behavior patterns as a reference-behavior pattern set for each legitimate subject. We mined behavior patterns from training data, and then incrementally merged these patterns to form an aggregate pattern set, which would be viewed as the reference-behavior pattern set for each subject.

Then for new operation sequences, we searched matches for each sequence based on the reference-behavior patterns, and then outputted all the matched patterns to represent the new data sequences. We considered a behavior segment from a new coming data sample as “behavior pattern” as long as it has a matching with one of the reference-behavior patterns.

4.4 Behavior-Pattern Analysis

Table 5 shows average ratios of the number of operations in different lengths of behavior patterns to the total number of operations in holistic behavior. The collected data for

TABLE 5
Ratio of Number of Operations in Different Lengths of Behavior Patterns to that in Holistic Behavior

Minimum support	Length-1 Pattern	Length-2 Pattern	Other Patterns	All Patterns
2%	22.17%	33.19%	37.48%	92.84%
5%	16.28%	28.74%	36.78%	81.80%
8%	13.87%	22.15%	30.82%	66.84%
20%	4.57%	3.34%	0%	7.91%

mining mouse behavior patterns consisted of about 4,800 mouse operations for each of 159 subjects. The ratios for different lengths of behavior patterns decrease as the minimum support increases. When the minimum support comes to 20 percent, the ratio for all patterns is just 7.91 percent. This may make the mined patterns provide no meaningful information about the behavior. In contrast, if we set the minimum support as a smaller value, for example, 2 percent in our study, it will generate more behavior patterns, but this may induce unstable and inconsistent mouse behavior, which may lead to lower authentication performance. Therefore, to obtain stable and applicable mouse behavior patterns, a balance should be made between the minimum support and the effectiveness of behavior patterns. In this study, we set the minimum support to be a trade-off value of 5 percent for usage, under which the corresponding ratio values for mined length-1, length-2 and length- n ($n \geq 3$) behavior pattern are 16.28, 28.74 and 36.78 percent respectively. Note that for easy presentation, we only present the ratio values in Table 5 at an average level. However, a similar observation holds for every subject.

5 FEATURE CONSTRUCTION

5.1 Feature Construction from Mined Patterns

To construct the feature vectors determining a users’ mouse behavior and validating his/her identity, procedural features are extracted from these patterns, which are organized into a vector to represent behavior patterns. We characterized mouse operations based on two basic properties: space and time. Each property was then analyzed individually, and converted into several features, to form the feature vector. This study defined seven fine-grained metrics to depict the behavior pattern, which can accurately and stably characterize a user’s unique mouse behavior (with feature dimensionalities in parentheses).

- *Click Elapsed Time:* we extracted time spent to perform a click action, including left/right single click and left/right double clicks. For single click, mean of overall time are extracted (2D); for double click, mean and standard deviation of one overall time and three interval times are extracted (8D).
- *Movement Speed:* we extracted pairwise-speed sequence by the rate of change of movement speed between two adjacent mouse events. Here we divided mouse movements into 8 types, depending on 4 movement directions (each expressing 90 degree ranges over 360 degree) and 2 movement distances (i.e., short and long) [6]. We then computed the ratio of travelled distance to movement duration

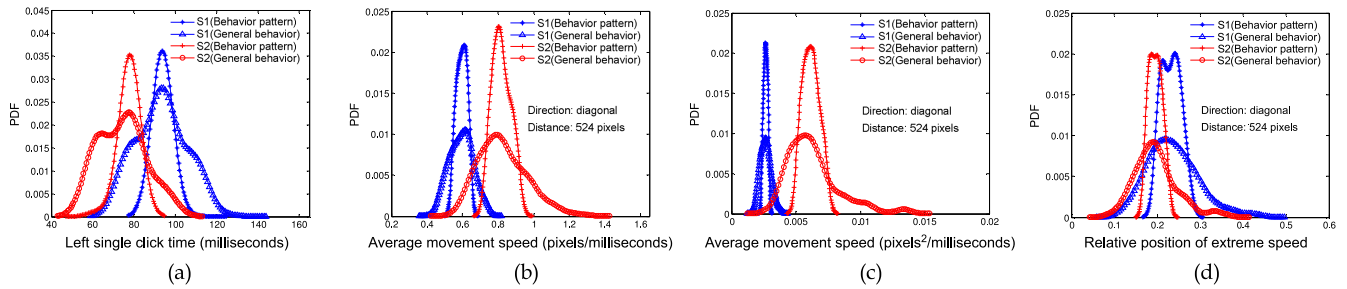


Fig. 2. Mouse features extracted from behavior pattern and holistic behavior for two different subjects (i.e., S1 and S2). Panel (a)-(d) show PDF curves of some typical features, including left single click time, average movement speed, average movement acceleration, and relative position of extreme speed.

time (1D), ratio of displacement to movement duration time (1D), and descriptive statistics¹ of the speed sequence (6D).

- *Relative-Position of Extreme Speed.* We calculated the relative position of the maximum speed value over the pairwise-speed sequence (1D). If the maximum speed value locates at the middle position of the speed curve, the relative position has the value of 0.5.
- *Movement Acceleration:* we computed descriptive statistics of the acceleration sequence (6D), which were obtained by the rate of change of speed sequence.
- *Movement Offset:* we first used coordinate transformation to place mouse movement at new coordinates, in which the X axis represents movement direction. We then obtained the movement offset by computing sum of coordinate values over the Y axis (1D).
- *Angle:* we obtained phase-angle sequences by computing angle between X axis and the line from screen origin to each mouse movement point, and the pairwise-angle sequence by computing the angle between X axis and the line from one movement point to the adjacent one. We computed descriptive statistics of these two sequences (12D).
- *Angle Speed:* we acquired angular-speed sequence by computing the rate of change of angle sequence between two movement points. We then computed descriptive statistics of the angular-speed sequence (6D).

It should be noted that there are three types of mouse movement in practice (see Table 2). But we only considered common movement and point-and-click movement, and chose to omit drag-and-drop movement. The reason is that this type of movement is rare for most subjects in our study, and has a large variability in our observation.

5.2 Empirical Feature Studies

5.2.1 Feature Stability in Behavior Pattern

One problem we came cross in analyzing mouse data was that the stability may be subject to behavioral variability. Features such as movement speed or acceleration are contingent upon the motion habits of individual mouse actions or the scenario in which a subject is under way. For example, a subject tends to move and click faster when he/she knows where the file is, and hesitates for a longer time if

he/she is trying to find that file. Therefore, this makes a good case to use the features from behavior patterns (containing application scenario information) for comparison between subjects.

We used a kernel density estimation method [13] to compute and compare the probability density function (PDF) of each mouse feature from behavior pattern and holistic behavior. Each feature's PDF is computed over 400 corresponding mouse operations from both the behavior pattern and the holistic behavior. Fig. 2 shows comparison of some typical features for two different subjects. The PDF curves for the features extracted from behavior patterns appear much more compact and concentrated than those from holistic behavior, which indicates that the characteristics in behavior patterns may allow one to more accurately and stably characterize mouse behavior. This may be due to frequent behavior segments providing more fine-grained and invariant information about mouse behavior, which also suggests that the features extracted from frequent behavior patterns are more stable and probably lead to a high authentication performance.

5.2.2 Feature Discriminability in Behavior Pattern

Another unique trait of the features from behavior patterns is that they are more distinctive among subjects than those from holistic behavior. Not only does the same subject have relatively stable feature values in behavior patterns, but different subjects have distinct feature values.

Fig. 2 shows that the PDF curves of the features from holistic behavior overlap with each other for two different subjects in a relatively large region, which makes it difficult to discriminate among subjects. As a comparison, there is a clearly distinctive gap between different subjects' PDF curves of features from behavior patterns, indicating features from behavior patterns hold more discriminative power. Along with the feature stability discussed before, this makes the features extracted from behavior pattern superior to those extracted from holistic behavior for discriminating between subjects. For easy presentation, we only compare the difference between a pair of subjects, but a similar observation holds for other subjects.

5.2.3 Statistical Dispersion of Features

Further to investigate the effectiveness of frequent behavior pattern, we defined a simple but effective dispersion metric based on Gini's Mean Difference [41]. The measurement of this metric is zero if all mouse feature values are identical, and increases as mouse feature values become diverse. For

1. Descriptive statistics quantitatively depict main features of a collection of data. Here we used 6 metrics for a mouse: mean, median, interquartile range, standard deviation, skewness, and kurtosis [46].

TABLE 6
Average Dispersion Metrics of Mouse Features for Holistic Behavior and Behavior Patterns

Features	Holistic Behavior	Behavior Pattern
Left single click time	0.1876	0.0175 (−90.67%)
Left double click time	0.0975	0.0204 (−79.08%)
Right single click time	0.1365	0.0298 (−78.17%)
Right double click time	0.1758	0.0614 (−65.07%)
Average movement speed	0.1957	0.0747 (−61.83%)
Average movement acceleration	0.3251	0.1317 (−59.49%)
Relative position of extreme speed	0.3742	0.1674 (−55.26%)

each feature f_k , with a sequence of values $\{x_i^k\}_{1 \times n_k}$, the dispersion metric $DM(f_k)$ is defined as,

$$DM(f_k) = \frac{1}{n_k(n_k - 1)} \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} |x_i^k - x_j^k| \quad (1)$$

We first computed the dispersion metric of each feature for every user over randomly selected 400 mouse operations (looping for 5 times). We then calculated the average dispersion metric for each feature over all 159 subjects.

Table 6 shows average dispersion metrics for some typical features extracted from behavior patterns and holistic behavior. The table also includes a percentage in parentheses, which indicates percentage decrease in dispersion metrics of behavior patterns over those of holistic behavior. By comparing the second and third columns, it is clear to see that features from behavior patterns are much more stable and discriminative than those from holistic behavior. Additionally, significant decreases of percentage are observed for all the dispersion metrics of features from behavior patterns. Specifically, almost all the metrics for behavior patterns decrease over 50 percent, and the decreased percentage for left single click time is up to 90.67 percent. This implies the features in behavior patterns have inherently unique attributes which are more stable and discriminative than the features in holistic behavior.

6 ACTIVE AUTHENTICATION ARCHITECTURE

The proposed approach consists of five modules (Fig. 3): behavior recorder, behavior-pattern extraction, feature construction, authentication model, and decision maker. This section focuses on the design of detector, the active authentication model, and that of the decision maker.

6.1 Design of One-Class Anomaly Detector

User authentication is typically a two-class (legitimate user versus impostors) problem, but in our context, usually only mouse data from the legitimate user are available in advance. Therefore, a practical solution is to build the authentication model on the data only from the legitimate user, and use the model to detect impostors, which is considered as a one-class learning problem [14].

6.1.1 Our Detector—One-Class Support Vector Machine

We employed a one-class Support Vector Machine (OC-SVM) [18], which has been applied to several biometric classification problems [36]. Given l training mouse feature

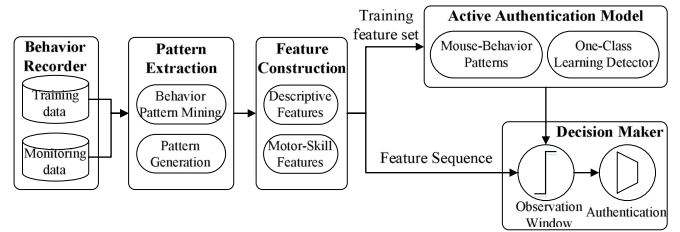


Fig. 3. System architecture of our active authentication approach.

samples $\{x_i \in \mathbb{R}^d\}$ belonging to one subject, $i = 1, \dots, l$, each sample has d features. To separate the data points from the origin, one needs to solve a dual quadratic programming problem [35]:

$$\max_{\beta} W(\beta) = \sum_{i=1}^l \beta_i \beta_j k(x_i, x_j) \quad (2)$$

$$\text{s.t.} \quad 0 \leq \beta_i \leq 1/v_l, \quad i = 1, \dots, l; \quad \sum_{i=1}^m \beta_i = 1 \quad (3)$$

where $\beta = \{\beta_i\}$ is the vector of l non-negative Lagrangian multipliers to be determined, v is a parameter that controls the trade-off between maximizing the number of data points contained by the hyperplane and the distance of the hyperplane from the origin, and $k(\cdot, \cdot)$ is a kernel function. Then the decision function

$$f(x) = \text{sign} \left(\sum_{i=1}^l \beta_i k(x_i, x_j) - \rho \right) \quad (4)$$

will be positive for the examples x_i from the training set, where ρ is the offset of the decision function.

The learning task was to build a detector based on a legitimate subject's feature samples. We used a radial basis function after a comparative study of linear, polynomial, and sigmoid kernels based on classification accuracy. The SVM parameter v and kernel parameter γ were set to 0.12 and 0.05 respectively.

6.1.2 Other Detectors

We implemented two other popular detectors in the literature—neural network and k -nearest neighbor (KNN) [42]. For neural network, a three-layer network was built with n input nodes, $2n + 1$ hidden nodes, and 1 output node, where n is feature dimensionality. During testing, the network output of a testing vector was considered as the detection score. For KNN, the covariance matrix of training feature samples was calculated. After multiple tests with k changing from 1 to 10, we obtained the best results with $k = 4$. In the testing phase, the average Mahalanobis distance from new feature sample to each sample in the training data was computed as the anomaly score.

6.2 Authentication Model

Mouse data are collected for a legitimate user whose behavior we are trying to model. We started by designating one subject as the legitimate user, and then built the authentication model for each legitimate user as follows.

Step 1: For training data, we obtained a set of behavior patterns as the reference-behavior patterns.

TABLE 7
FARs for Behavior Pattern and Holistic Behavior at Operation Length of 16 (standard deviation in parentheses)

Detector	Behavior Pattern	Holistic Behavior
	FAR (%) @ FRR = 1%	FAR (%) @ FRR = 1%
Nearest Neighbor	2.87 (1.24)	12.19 (7.37)
Neural Network	0.69 (0.05)	9.57 (5.32)
One-Class SVM	0.09 (0.03)	7.27 (3.25)

Step 2: We extracted features from each behavior pattern to form a feature vector.

Step 3: We took these feature vectors as the training set to train our detector, and to build the authentication model.

6.3 Decision Maker

After building the authentication model of normal behavior we exploited this model to verify whether current behavior is normal. The legitimate user's profile, obtained at the training stage, is used to compare with current user's mouse operations. If there is a significant difference, the user is considered as an anomalous one. Specifically, for a new mouse sequence, we first extracted matched patterns given the reference-behavior patterns, from which the feature vector was obtained. We also employed an observation window to make authentication decision, which contained a sequence of N mouse operations. We adopted a buffer authentication mechanism, which could effectively decrease error rates.

7 USER STUDY AND EVALUATION METHODOLOGY

7.1 Data Collection

7.1.1 Data Collection

A free experimental environment was established to collect mouse behavior data in this study. We developed a data collection software that runs as a background job, and starts monitoring a subject's actions when his/her login occurs and stops running when his/her logout occurs; the software is transparent and does not affect other applications. Mouse data were collected during subject's routine computer activities, which were mainly under applications of Internet surfing, word processing, game playing, online chatting, and programming. Whenever a subject moved or clicked the mouse, the software recorded action type (e.g., move or click), position at which the action occurred, application information in which the action occurred, and timestamp when the action occurred. The Windows clock is used to construct the timestamp, which has a resolution of ± 15.625 milliseconds.

7.1.2 Platform

We asked the subjects to use their own computers for data collection, and the collected data were sent to a remote server via the Internet periodically. The server stores the data in an internal database, along with the subject ID. The desktops are Lenovo and HP workstations, which are equipped with 17" HP LCD monitors (set at 1280×1024 resolution), and USB optical mice, running the Windows

XP or Windows 7 system. The server configuration is a Dell PowerEdge server with an Intel Xeon X5677 3.46 GHz Quad Core Processor with 12.0 GB of RAM, running the Windows Server 2008 operating system.

7.1.3 Running Subjects

We recruited 159 graduate students and faculties (61 females and 98 males) from the university, with ages ranging from 16 to 52 years (mean = 26.8, s.d. = 7.9). All participants were skilled mouse users with at least one year's experience, and thirteen of them were left-handed. All subjects performed in their routine computer usage for the data acquisition. We collected around two weeks' data of routine mouse usage during their daily worktime. Each subject contributed around 9,600 mouse operations (mean = 9,425, median = 9,174, and s.d. = 677.8).

7.2 Evaluation Methodology

7.2.1 Training and Testing Procedure

We designated one of our subjects as the legitimate user, and the rest as impostors. We trained and tested each detector to identify the legitimate user and impostors:

Step 1: We ran the training phase of the detector on the first two days of data from the legitimate user to train his/her authentication model. The training data size for each subject is around 1,800 mouse operations (mean = 1,777, median = 1,703, s.d. = 368).

Step 2: We ran the testing phase of the detector on the remaining data from that user.

Step 3: We ran the testing phase of the detector on the data from all impostors.

This process was repeated by designating each remaining subjects as the legitimate user in turn. For choosing parameters, 10-fold cross validation [42] was employed.

7.2.2 Calculating Detection Performance

To evaluate the performance of the proposed approach, we calculate the false-acceptance rate (FAR) and false-rejection rate (FRR). Specifically, the FAR is computed as the ratio between the number of false acceptances and the number of test samples from impostors; the FRR is computed as the ratio between the number of false rejections and the number of test samples from legitimate users. We also brought FAR and FRR together to generate a graphical summary of performance known as ROC curve [42]. Whether or not a feature sample produces an alarm depends on how the threshold on the detection score is chosen. We chose the default threshold of 0.0 to calculate FAR and FRR, and computed equal-error rate (EER) for the detector's sensitivity when FAR equals FRR.

8 RESULTS AND ANALYSIS

8.1 Active Authentication

8.1.1 Authentication Performance

Table 7 and Fig. 4 show average EERs and FARs (with fixed FRR of 1 percent due to the European Standard [9]) of active authentication for holistic behavior and behavior patterns, with standard deviation in parentheses.

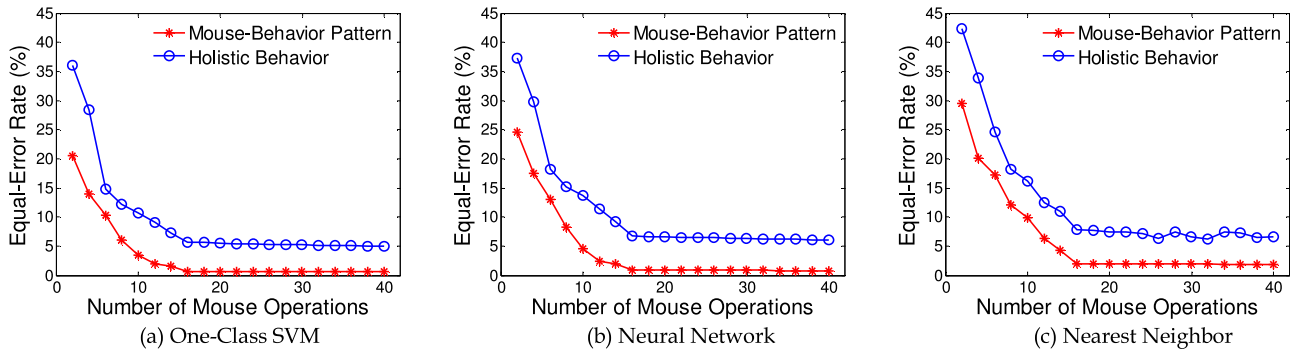


Fig. 4. EER curves for mouse behavior patterns and holistic mouse operations at varying operating lengths using three types of detectors. X-axis represents the length of mouse sequence used to verify a user's identity.

The best performance has a FAR of 0.09 percent with FRR of 1 percent, which is impressive and achieves an acceptable level of accuracy for practical application. It is also competitive with the best results previously reported, while being subject to more behavioral variability compared with previous work, since they observed mouse activities in a longer period. Moreover, average error rates for features from the holistic behavior are much higher than those from behavior patterns, which are probably due to better compactness and less variability of mouse behavior by using behavior patterns. Although it might be unwise to rely solely on an authentication method which is not up to the European standard [9] (with FAR of 0.001 percent and FRR of 1 percent), it does indicate that mouse behavior could provide evidence in the active authentication task. Also, corresponding standard deviations of the features from behavior patterns are much smaller than the features from holistic behavior, suggesting that behavior patterns might be more stable and robust to behavioral variability.

One-class SVM performs much better than other detectors. It can capture the density of a hypersphere of normal behavior, and its support vectors enable it to detect outliers where the data are not linearly separable, and to find informative features with relative insufficiency of prior knowledge [43]. The smaller standard deviations indicate the detector is relatively robust to behavioral variability.

Additionally, we conducted a statistical test using confidence interval and half-total error rate (HTER) [22], to further statistically analyze our performance. Our approach with one-class SVM provides the lowest HTER; the 95 percent confidence interval lies at 0.63 ± 0.35 percent.

8.1.2 Usability to Operation Length

Operation length refers to the amount of mouse operations for authentication, and denotes the observation time for authentication. When deciding with five mouse operations, the EER is approximately 15 percent, but the authentication decision only needs 8.77 seconds (on average). All detectors obtain smaller error rates when increasing the number of operations for authentication decision. As the operation length increases to 20, the best EER drops to 0.75 percent, and the corresponding time increases to 47.11 seconds. Therefore, this introduces a tradeoff between authentication accuracy and required time to make authentication decision, and this time has a direct effect on the security since it says how long an attacker can interact with the devices.

Also it is notable that all EERs almost converge at a level of 15 to 18 operations (corresponding to an average time of 39.13 seconds), and stay there (with only small fluctuations) up to using 40 operations.

8.1.3 System Implementation and Overhead Analysis

For one hand, this authentication system can be used for user authentication on a personal or public computer in an organization. The system is installed at the client side, and usually, a single subject is presented for authentication. Thus, the end host can easily fulfill the authentication task; the system load is very light. On the other hand, such a system can be integrated into online services (e.g., e-account authentication), where the data collection module is embedded within a user's account page, and the analysis module is placed on the server side. Since hundreds of users might simultaneously access their accounts, the system overhead may become an issue in terms of computational cost and storage resources.

We evaluated computational overhead for behavior-data processing, feature extraction, and detection. The CPU cost was measured on an Intel Xeon X5677 3.46-GHz Processor. The data processing and feature extraction for 5 minutes' mouse inputs from 100 users, including around 9110 mouse operations, take only 16.57 s; and the detection takes only 872 ms. These modules were processed in an encrypted domain (using the standard encryption technique of AES with key length of 256 bits); the time consumption was 957 ms. Compared with the data processing and feature extraction tasks, the overhead for detection and data protection are negligible, which indicates that computational overhead is minor.

We also assessed the memory cost of the authentication system. The authentication process was analyzed using a Windows tool, *purify*², only 87.97 Kb of memory resource was consumed for testing a 5 minutes' mouse-operation sequence. Our system prototype was developed to utilize a single threaded with multiple client modes with time-division multiplexing. The primary memory cost is to buffer accumulative mouse operations and detector outputs for each user. A single mouse operation consumes 252 bytes, 4 bytes for each feature metric. A detection sequence of 20 mouse operations consumes 4.92 Kbytes. If the user space increases to 100 online users, it requires 492.34 Kbytes in total. Another issue of concern is the disk space for storing

2. https://en.wikipedia.org/wiki/Rational_Purify

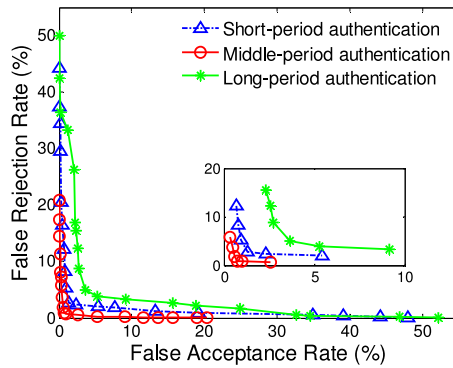


Fig. 5. ROC curves of active authentication across three scenarios

user profiles. In our system, a user’s behavior template consumes 497.14 Kbytes. If the user space increases to 10,000 users, it requires 3.95 GB, which is relatively light, even on a personal computer.

8.2 Reliability to Application Scenario

Reliability and applicability of the proposed approach also vary with application scenarios [17], [44]. In real life, users may have different usage frequencies of computer mouse under different scenarios. Users may frequently use their mouse in several short periods; or intermittently use their mouse during a day. Here we considered three application scenarios; each corresponds to a way of mouse behavior for authentication.

Short-Period Scenario. A user operates computer mouse for a while, and then gets away from computer. The computer then starts to build the user’s mouse-behavior profile. After several minutes or hours, the user logs onto the computer, and the computer turns to the detection mode. This scenario allows users to use the computer for a longer time without locking or unlocking the computer. Here we separated the data of the first day into three parts evenly, and selected the first part as the training data, and set the third as the testing data. The training data size for each subject is around 390 mouse operations ($mean = 381, s.d. = 38.5$).

Middle-Period Scenario. A user’s profile and model are built on the mouse operations for a relatively long observation time. The model stays the same for a relative-long time up to few days, and then detects the legitimacy of users. Here we chose the data of the first two days for model training, and used the data of the fourth and fifth days for model testing (in which the data are captured about two days later). The size of training data for each subject is around 1,800 mouse operations ($mean = 1,777, s.d. = 398$).

Long-Period Scenario. We used the same methodology and training data in the middle-period scenario to build the authentication model. The model stays the same for a longer time up to several days, and then detects the identity of users. We trained the model on the data of first two days, and test it on the data of the ninth and tenth days (in which the data are captured seven days after).

We employed the one-class SVM detector to conduct authentication experiments with different timespans between enrollment and authentication, and specified the operation length of 16. Fig. 5 shows ROC curves across three application scenarios. The middle-period scenario has the best performance, and the performance in the short-period

TABLE 8
FARs for Behavior Pattern and Holistic Behavior at Operation Length of 16 (standard deviation in parentheses)

Application Tasks	Behavior Pattern	Holistic Behavior
	FAR(%) @FRR = 1%	FAR (%) @FRR = 1%
Web surfing	0.05 (0.02)	4.78 (2.24)
File finding	0.03 (0.01)	3.14 (1.52)
Word processing	0.08 (0.03)	7.05 (3.16)
Free task	0.09 (0.03)	7.27 (3.25)

scenario is relatively better than that in the long-period scenario. Specifically, the FRR and FAR are 0.89 and 0.73 percent in the middle-period scenario; while in the short-period scenario, the FRR and FAR increase to 2.74 and 1.28 percent. The reason is probably that there has a longer observation of mouse behavior in the training phase of the middle-period scenario, leading to more stable and accurate user profiles. The FRR and FAR in the long-period scenario are 5.04 and 3.62 percent, which is a little worse than that in the short-period scenario. These may be due to the larger timespan between training and testing phases in the long-period scenario, which could introduce a long period of behavioral variability.

8.3 Sensitivity to Application Task

This experiment examined the performance across different application tasks. We chose four commonly used application tasks: web surfing, file finding, word processing, and free task. Each application task reflects some typical operation habits in users’ daily computer activities. We derived the datasets of different application tasks based on the application-type information. We had 92 subjects in this experiment because these subjects had the data over all four tasks. We applied one-class SVM detector, and set the number of mouse operations for decision as 16.

Table 8 shows the average FARs at a fixed FRR (1 percent) across different application tasks. The error rates are lower than 1 percent in every application task, which indicates mouse behavior could offer useful information for active authentication across different application tasks. Besides, the authentication in a specific task performs better than those in the free task. Specifically, the FAR obtained in the free task is 0.09 percent when the FRR is 1 percent, while in the task of web surfing, the FAR reduces to 0.03 percent when FRR remains to be 1 percent. The reason is probably due to the avoidance of inter-task variability in a specific task. Thus we can conclude that users’ mouse behaviors exhibit less variability in a specific application task than that in the unconstrained one, and may lead to a better performance.

8.4 Scalability to User Space

Scalability of user space denotes the ability of mouse behavior being enlarged to accommodate a growth of user space. It is usually true that with an increasing user space, there is a higher chance that two users have similar profiles. Thus one wonders how the space fills in as the number of users increases. Here we set the size of user space to be a variable ranging from 2 to 50. Specifically, we used the free-task scenario as the test bed, and for each such variable, we repeat

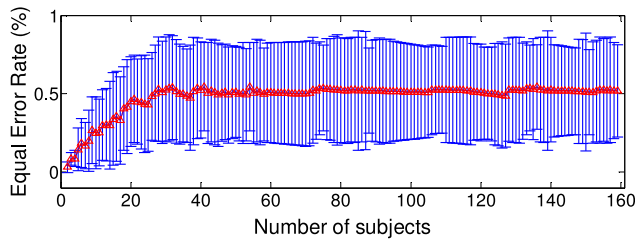


Fig. 6. EERs of active authentication at varying sample length.

the evaluation twenty times with a randomly selected collection of users.

The EERs with different sizes of the user space, together with their standard deviations, are presented in Fig. 6. The authentication error rates increase as the user space becomes larger. Specifically, there is a significant increase of the error rate in the interval between 2 and 22 users. This is as expected, since larger user space usually means a higher probability that two users have similar profiles. Also, we observe that when the user size is larger than about 25 users, the authentication error rates become relatively stable, and only small fluctuations with the error range are apparent. These results indicate that the user size should be (at least) larger than 25, in which case the influence of the user space may be minimal. These results also indicate our subject size is located in a range where the influence could be negligible.

8.5 Robustness to Mimic Attack

Theoretically, our authentication system can be bypassed if an impostor can precisely mimic mouse motions of the computer's owner (e.g., close friends or workmates). However, this is extremely difficult (if not impossible) in practice. Intuitively, even if the impostor has overseen how the owner operates the mouse, it might be able to mimic the operation aspect. But the other habit or behavior features, such as speed and acceleration, are more difficult to observe and reproduce. To quantitatively measure the effect of mimic by observation, we set up an experiment involving additional 23 users. Each user is first considered as the target user and is observed closely by the remaining users who try to mimic the target user's mouse operations. 8 users have the same gender and similar body size as the target user, while other 15 have different gender and body size than the target user. Before mimicking, four impostors closely observe how the target user performs mouse operations. Table 9 shows average FARs and FRRs of the mimic attacks. Compared with the results in Table 8, there is no significant improvement in mimicking given the observation, and it is evident that a mimic attack is hard to succeed.

8.6 Response to Behavior Change

This work builds on the assumption that a user's behavior is consistent and no abrupt change happens over a period of time, but the assumption might not always be true, e.g., due to physical injuries of hands. In such cases, the authentication mechanism should stay minimally intrusive to the user. One feasible solution is to contact with the service managers to remotely disable the authentication function and start the re-training. As we have shown previously, the sensitivity to false positives and negatives are controlled by various thresholds. Whether or not exposing the sensitivity

TABLE 9
FARs and FRRs of Mimic Attacks for Active Authentication
(standard deviations in parentheses)

Detector	Short-period scenario (%)		Middle-period scenario (%)		Long-period scenario (%)	
	FAR	FRR	FAR	FRR	FAR	FRR
One-Class	1.57	2.91	0.75	0.88	3.71	5.12
SVM	(0.91)	(1.18)	(0.29)	(0.43)	(1.67)	(2.83)

control (e.g., setting it to Low, Medium, and High) can improve user experience is debatable. On one hand, it allows users to make a conscientious choice to tradeoff between security and usability. On the other hand, it is no longer user-transparent.

8.7 Comparison with Previous Work

Most researchers built their models on different algorithms and datasets, but few of them made informed comparisons among different algorithms and results. Here two comparative experiments are provided to compare our approach with the state-of-the-art approaches.

8.7.1 Comparison with Existing Approaches

We first quantitatively compared our approach with the state-of-the-art approaches using the same dataset and test protocol: Pusara & Brodley [1], Gmaboa & Fred [5], Ahmed & Traore [6], Shen et al. [38], Mondal and Bours [33], Rahman et al. [31], and Zheng et al. [32]. These seven approaches represent a relatively diverse set of mouse features and algorithms. To implement these approaches, we tried to faithfully choose features and algorithms used in the approaches. The evaluation methodology is same as the one in Section 7.2.1, with the operation length of 16.

Table 10 shows the average FARs at a fixed FRR (1 percent) for seven approaches. When all approaches are evaluated under the same condition, our approach performs the best. This may be due to our frequent behavior patterns in obtaining stable mouse characteristics, which generates a more discriminative and stable representation of mouse behavior, thus leads to a boost in authentication performance. The standard deviations of error rates for our approach are smaller than those of other approaches, suggesting that our approach may be more robust to behavioral variability. We tried to faithfully implement these approaches, but we note ambiguities may inadvertently cause our reimplementations to differ from the originals. Hence the results may also be due, in part, to: (1) collecting data in different scenarios; (2) using a data set that contains more variability and our approach targeting to this problem; (3) adopting different parameters of authentication algorithms; (4) using less enrollment data to train the detector than that used in other studies; (5) using different types of thresholds on detection scores.

The results only provide preliminary comparative results and should not be concluded that a certain algorithm is always better than others. Each algorithm has its own advantages and disadvantages under different conditions, so further comparisons on more realistic and diverse datasets are needed.

TABLE 10
Comparison with Existing Approaches (Standard Deviations in Parentheses)

Source Study	Feature Source	Detectors	FAR (%) @ FRR = 1%	Settings
Pusara & Brodley (2004) [1]	Click & Movement	Decision Tree	18.71 (10.34)	Continuous
Gmaboa & Fred (2004) [5]	Movement	SFS + Weibull Distribution	16.43 (9.18)	Continuous
Ahmed and Traore (2007) [6]	Movement	Neural Network	11.25 (6.71)	Continuous
*Shen et al. (2013) [38]	Movement	PCA + SVM	15.93 (8.41)	Static
Mondal and Bours (2013) [33]	Movement	SVM	9.73 (4.89)	Continuous
Rahman et al. (2015) [31]	Click & Movement	Outlier Filtering and Counting	29.74 (13.57)	Continuous
Zheng et al. (2016) [32]	Movement	SVM	6.13 (2.97)	Continuous
Our study	Click & Movement	PrefixSpan + One-Class SVM	0.09 (0.03)	Continuous

*: The approaches in [38] are initially applied to static authentication. The reason for this choice is that we want to examine whether the approaches employed in static authentication can be used in continuous authentication.

8.7.2 Comparison with Previous Work

Table 11 presents a qualitative comparison of our experimental settings and results against previous work for active and continuous authentication. Pusara et al. [1] obtained an acceptable accuracy for authentication, but the user size (11) seems to be insufficient to obtain steady results. Schulz [8] and Rahman et al. [31] may make the authentication decision in a relatively short time, but the error rates were up to 24.3 percent [8] and 13.42 percent [31], which may lead to an unreliable mechanism in practice. Ahmed et al. [6], Nakkabi et al. [40] and Mondal & Bours [33] could achieve a very high accuracy, but the required data for authentication were relatively large. Additionally, most existing work used data from both legitimate user and impostors for training the authentication model, which may be not realistic in practice, since usually only the data from legitimate user are readily available in practical scenarios.

However, our study relied on an improved authentication methodology using frequent behavior patterns, analyzed mouse behavior from perspectives of its discriminative power and behavior variability, and had a larger subject pool, a more carefully chosen set of behavioral metrics, more extensive experiments than previous work did, which leads to a good performance. We established one-class learning model with which only data from legitimate user were needed to build authentication model. We achieved a high authentication accuracy of 0.09 percent FAR and 1 percent FRR, with a short authentication time. We also implemented

our approach into a current operation system which shows light computational and memory cost.

9 DISCUSSIONS

We developed a simple but efficient authentication system through modeling this behavior. Experimental results show the discriminability and stability of mouse behavior could be significantly improved by behavior patterns, and can achieve a FAR of 0.09 percent and FRR of 1 percent in some cases. But it is still not low enough to meet the European standard for commercial biometrics. Thus further progress is needed before we can depend solely on mouse behavior for active authentication. One way to improve the accuracy is to seek more accurate and stable representation of mouse behavior. Other ways may be to reduce noise in mouse data or employ multiple detectors for separately depicting various properties of mouse behavior.

We modeled mouse behavior by proposing a pattern-growth-based mining method to extract frequent behavior patterns in obtaining stable mouse characteristics. Empirical studies show a noticeable difference among users' behavior when using the features from behavior patterns, and indicate the behavior patterns can greatly reduce the variability, which would lead to a performance improvement. Additionally, we employed a one-class learning method to build the authentication model, so that the model can be trained solely on the samples from the legitimate user. An in-depth

TABLE 11
Qualitative Comparison with Existing Work For Active and Continuous Authentication

Source Study	Authentication Results			Data Collection		Training Data Source
	FAR	FRR	Data Required	Users	Action	
Pusara and Brodley (2004) [1]	0.43%	1.75%	Not specified	11	Click & Movement	Legal user and Impostors
Gmaboa and Fred (2004) [5]	2%	2%	50 mouse strokes	50	Movement in a game	Legal user and Impostors
Schulz (2006) [8]	24.3%	24.3%	Not specified	72	Movement	Legal user and Impostors
Ahmed and Traore (2007) [6]	2.46%	2.46%	2000 mouse actions	22	Movement	Legal user and Impostors
Nakkabi et al. (2010) [40]	0%	0.36%	2000 mouse actions	48	Movement	Legal user and Impostors
Zheng et al. (2011) [25]	1.3%	1.3%	37.37 minutes	30	Movement	Legal user and Impostors
Shen et al. (2012) [39]	0.37%	1.12%	3000 mouse actions	28	Click & Movement	Legal user and Impostors
Mondal and Bours (2013) [33]	-	0%	94 mouse actions	49	Movement	Legal user and Impostors
Rahman et al. (2015) [31]	13.42%	13.42%	Not specified	45	Click & Movement	Not specified
Zheng et al. (2016) [32]	1.3%	1.3%	20 clicks*	30	Movement	Legal user and Impostors

*: If they choose 20 mouse clicks in a detecting block, the verification time could be 37.73 minutes; however, the verification time will be reduced to 3.03 minutes if partial movements are used.

and comprehensive survey on other detectors is planned to help progress toward better performance.

We examined authentication performance across various application tasks. The authentication accuracies in a specific task perform better than those in free task, which is probably due to the fact that the mouse-behavior variability between application tasks is larger than that within a specific application task. To mitigate the variability introduced by the behavior difference between application tasks, one could consider context information of mouse operations as an additional feature source [27], or respectively build a user's profile in each application task. These ways may also extend the generalization capability of this biometric in practice.

We explored active authentication under various application scenarios. The application scenarios, in which the mouse behavior is observed in a longer period of time or the authentication model has a smaller timespan between its training and detection modes, would produce better performance. Thus further processes are required to extend the flexibility of this approach. One possible way to address this problem is to employ effective online incremental learning strategies [28] to continuously learn new mouse operations, and to increasingly enhance validity of the authentication model.

For user authentication through mouse behavior, one important thing is that the registration and authentication process should take place in a protected environment, to avoid impostors maliciously manipulating users' profile data. In our system implementation, we considered a simple way to secure the profile data by storing it in an encrypted domain (using standard encryption techniques of AES), but this may lead to the risk of leaving the profile data exposed during each authentication. A securer way is to employ a transformation function to the profile data [29], and only store the transformed profile data. Besides, during the profile generating phase, users' identity should be checked in an alternative way, and the production of mouse-behavior samples should be limited to a few days, not to weeks or months.

When behavior-based techniques are utilized for user authentication, it may raise concerns about user's privacy. At least users should be aware that they are under observation, and also understand that every security policy must imply a limitation of their privacy in some way [30]. Compared with other behavior-based authentication methods (e.g., keystroke dynamics may record users' passwords and sensitive texts), the personal information embedded in mouse behavior is minimal. Mouse behavior analysis monitors users' mouse movements and clicks, and will have to be blurred for the privacy concern. Mouse operations would be normalized to have relative position information and be only stored in terms of features, which are available exclusively to authentication process. Even with perfect knowledge of stored mouse operations, the only thing an adversary may figure out is when the user clicked or moved the mouse, which gives away little information about users' credentials.

This study has shown promising active authentication performance in a homogeneous computing environment, but in more practice we are aware that a user may use quite different computing platforms, which may lead to distributional differences of mouse-behavior features across different computing platforms. Real-world distributional difference across different computing environments often comes from

(1) hardware-level factors (e.g., computer type); (2) software-level factors (e.g., screen resolution); (3) environmental factors (e.g., distance between monitor and body); and (4) psychological and physiological state of the subject (e.g., the subject may be fatigued or distressed). Thus it is reasonable to wonder whether these factors would change mouse data if the states of these factors are different at different environments (e.g., two quite different working platforms) or moments (e.g., enrollment time versus authentication time). This problem where the data for model training have a different distribution from the data for decision making is often known as domain adaptation [45]. One such method based on faces and touch gestures for smartphone authentication using domain adaptation was recently proposed in [46]. Domain adaptation and transfer learning techniques can be used to deal with the changing distribution problem in continuous authentication [47], which deserves more attention in future work.

Implicit and continuous data capture of mouse behavior makes it appropriate for active and continuous user authentication. For practical application of this approach, mouse-behavior data were captured implicitly and passively during users' routine computer usage, and a threshold was set to trigger the establishment of authentication model for a legitimate user. When the number of mouse operations reached a pre-set threshold, authentication model will be built. A periodic model updating strategy would be introduced for robust and accurate authentication. Besides, unlike other approaches in this field, our authentication decision is made on a certain length of mouse operations instead of basing on an operation block, and can achieve an acceptable performance of near-real-time continuous authentication. However, if one wants to apply this method to a real-time authentication (which is updated with each mouse operation), what features should be used, and how mouse behavior can be integrated efficiently with the current authentication mechanisms are still open questions, and will be investigated in our future work.

10 CONCLUSION

Mouse-interaction behavior analysis is a newly emerging behavioral biometric, which offers a capability of verifying a user's identity in a transparent and continuous manner. In this study, we highlighted motivation and challenges of using mouse behavior for active authentication, and developed a simple but efficient approach that can perform the active authentication task in a short time while maintaining high accuracy. We presented a novel approach by using pattern-growth based mining method to extract behavior patterns, employing a one-class learning algorithm to detect abnormal behavior. By constructing a mouse behavior data set under a realistic environment, we examined the efficacy of our proposed approach through a series of experiments across various application scenarios and tasks. Additionally, we explored the proposed approach in terms of usability to sample length, robustness to mimic attack, and response to user behavior change, to further examine the applicability of the approach. We also performed two comparative experiments to compare our approach with the state-of-the-art approaches.

The experimental results show that the mouse characteristics from behavior patterns are much more stable than

those from holistic behavior. The performance is competitive and reaches a practically useful level for realistic deployment. These findings suggest that mouse-interaction behavior is sufficient to be an important enhancement for a traditional active/continuous (re)authentication system. Although a large amount of work has been done in this area, many issues remain open such as variability reduction and online learning of behavior detectors. Further investigations in more realistic settings are planned.

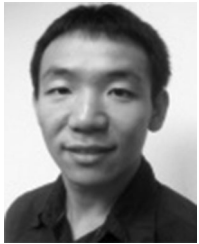
ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China (61403301, 61773310), China Postdoctoral Science Foundation (2014M560783), Special Foundation of China Postdoctoral Science (2015T81032), Natural Science Foundation of Shaanxi Province (2015JQ6216), Open Project Program of the National Laboratory of Pattern Recognition (NLPR) and Fundamental Research Funds for the Central Universities (xj2015115). Roy Maxion was supported by National Science Foundation of United States (CNS-1319117).

REFERENCES

- M. Pusara and C. E. Brodley, "User re-authentication via mouse movements," in *Proc. ACM Workshop Visual. Data Mining Comput. Secur.*, 2004, pp. 1–8.
- P. Bours and C. J. Fullu, "A login system using mouse dynamics," in *Proc. Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, 2009, pp. 1072–1077.
- Defense Advanced Research Projects Agency, "Active authentication Announcement," Defense Advanced Research Projects Agency, Arlington County, VA, USA, Tech. Rep. DARPA-BAA-12-06, 2012.
- R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: A survey and classification," *J. Biometrics*, vol. 1, no. 1, pp. 81–113, 2008.
- H. Gamboa and A. Fred, "A behavioral biometric system based on human computer interaction," *Proc. SPIE*, vol. 2004, pp. 4–26, 2004.
- A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Trans. Dependable Secure Comput.*, vol. 4, no. 3, pp. 165–179, Jul. 2007.
- Z. Cai, C. Shen, and X. Guan, "Mitigating behavioral variability for mouse dynamics: A dimensionality-reduction-based approach," *IEEE Trans. Human-Mach. Syst.*, vol. 44, no. 2, pp. 244–255, Apr. 2014.
- D. Schulz, "Mouse curve biometrics," in *Proc. Symp. Biometrics*, 2006, pp. 1–6.
- CENELEC, "European Standard EN 50133-1: Alarm systems. Access control systems for use in security applications. Part 1: System requirements," European Committee for Electrotechnical Standardization, Standard Number EN 50133-1:1996/A1:2002, 2002.
- F. Bergadano, D. Guneti, and C. Picardi, "User authentication through keystroke dynamics," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 4, pp. 367–397, 2002.
- H. Gamboa and A. Fred, "An identity authentication system based on human computer interaction behavior," in *Proc. Int. Wrok. Pattern Recognit. Inf. Syst.*, 2003, pp. 46–55.
- S. Mondal and P. Bours, "A study on continuous authentication using a combination of keystroke and mouse biometrics," *Neurocomput.*, vol. 230, pp. 1–22, 2017.
- A. W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis*. New York, NY, USA: Oxford Univ. Press, 1997.
- D. Tax, "One-class classification; concept-learning in the absence of counter-examples," *Ph.D. thesis*, Delft University of Technology, Delft, Netherlands, 2001.
- K. Mickelberg, N. Pollard, and L. Schive, "US cybercrime: Rising risks, reduced readiness Key findings from the 2014 US State of Cybercrime Survey," US Secret Service, National Threat Assessment Center, Pricewaterhousecoopers, 2014.
- Z. Jorgensen and T. Yu, "On mouse dynamics as a behavioral biometric for authentication," in *Proc. ACM Symp. Inf. Comput. Commun. Secur.*, 2011, pp. 476–482.
- V. Matyáš and Z. Říha, "Biometric authentication—security and usability," in *Proc. Adv. Commun. Multimedia Secur.*, 2002, pp. 227–239.
- C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, 2011, Art. no. 27.
- J. Pei, et al., "Mining sequential patterns by pattern-growth: The PrefixSpan approach," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 11, pp. 1424–1440, Nov. 2004.
- R. Kaminsky, M. Enevand, and E. Andersen, "Identifying game players with mouse biometrics," University of Washington, 2008.
- R. Everitt and P. W. McOwan, "Java-based internet biometric authentication system," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1166–1172, Sep. 2003.
- S. Bengio and J. Mariethoz, "A statistical significance test for person authentication," in *Proc. Odyssey*, 2004, Art. no. 237.
- S. Mondal and P. Bours, "A computational approach to the continuous authentication biometric system," *Inf. Sci.*, vol. 304, pp. 28–53, 2015.
- Australian Cyber Security Centre, 2015 Cyber Security Survey, 2015. [Online]. Available: <https://www.cert.gov.au/system/files/614/691/2015-ACSC-Cyber-Security-Survey-Major-Australian-Businesses.pdf>
- N. Zheng, A. Paloski, and H. M. Wang, "An efficient user verification system via mouse movements," presented at the *Proc. ACM Conf. Comput. Commun. Secur.*, Chicago, USA, 2011.
- Australian Cyber Security Centre, 2015 Cyber Security Survey, 2016. [Online]. Available: https://www.acsc.gov.au/publications/ACSC_Threat_Report_2016.pdf.
- T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi, "TIPS: context-aware implicit user identification using touch screen in uncontrolled environments," in *Proc. Workshop Mobile Comput. Syst. Appl.*, 2014, pp. 1–6.
- C. Lin, Y. Song, and Z. Wen, "Sequential anomaly detection," United States patent application US 13/969,151. Aug. 16, 2013.
- A. Rua, E. Maiorana, J. Castro, and P. Campisi, "Biometric template protection using universal background models: An application to online signature," *IEEE Trans. Information Forensics and Security*, vol. 7, no. 1, pp. 269–282, Feb. 2012.
- K. Nandakumar and A. K. Jain, "Biometric template protection: Bridging the performance gap between theory and practice," *IEEE Signal Process. Mag.*, vol. 32, no. 5, pp. 88–100, Sep. 2015.
- K. Rahman, R. Moormann, D. Dierich, and S. Hossain, "Continuous user verification via mouse activities," in *Proc. Int. Conf. Multimedia Commun. Serv. Secur.*, 2015, pp. 170–181.
- N. Zheng, A. Paloski, and H. M. Wang, "An efficient user verification system using angle-based mouse movement biometrics," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 3, pp. 1–27, 2016.
- S. Mondal and P. Bours, "Continuous authentication using mouse dynamics," in *Proc. Int. Conf. Biomet. Special Interest Group*, 2013, pp. 1–12.
- B. Sayed, I. Traoré, I. Woungang, and M. Obaidat, "Biometric authentication using mouse gesture dynamics," *IEEE Syst. J.*, vol. 7, no. 2, pp. 262–274, Jun. 2013.
- B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.
- A. K. Jain, A. Ross, and S. Pankanti, "Biometrics: A tool for information security," *IEEE Trans. Inf. Forens. Secur.*, vol. 1, no. 2, pp. 125–143, Jun. 2006.
- A. Syukri, E. Okamoto, and M. Mambo, "A user identification system using signature written with mouse," in *Proc. Australasian Conf. Inf. Secur. Privacy*, 1998, pp. 403–414.
- C. Shen, Z. Cai, X. Guan, Y. Du, and R. Maxion, "User authentication through mouse dynamics," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 1, pp. 16–30, Jan. 2013.
- C. Shen, Z. M. Cai, and X. H. Guan, "Continuous authentication for mouse dynamics: A pattern-growth approach," in *Proc. IEEE/IFIP Conf. Dependable Syst. Netw.*, 2012, pp. 1–12.
- Y. Nakkabi, I. Traore, and A. A. E. Ahmed, "Improving mouse dynamics biometric performance using variance reduction via extractors with separate features," *IEEE Trans. Syst. Man Cybernetics Part a-Syst. Humans*, vol. 40, no. 6, pp. 1345–1353, Nov. 2010.

- [41] S. Yitzhaki, "Gini's Mean difference: A superior measure of variability for non-normal distributions," *Metron-Int. J. Statistics*, vol. 61, no. 2, pp. 285–316, 2003.
- [42] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. Hoboken, NJ, USA: Wiley, 2001.
- [43] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics Comput.*, vol. 14, no. 3, pp. 199–222, 2004.
- [44] C. Braz and J. M. Robert, "Security and usability: The case of the user authentication methods," in *Proc. 18th Conf. Interaction Homme-Mach.*, 2006, pp. 199–203.
- [45] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 53–69, 2015.
- [46] H. Zhang, V. M. Patel, S. Shekhar, and R. Chellappa, "Domain adaptive sparse representation-based classification," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recognit.*, vol. 1, pp. 1–8, 2015.
- [47] I. Žliobaitė, "Learning under concept drift: An overview," *CoRR*, vol. abs/1010.4784, 2010.



Chao Shen (S'09-M'14) is currently an associate professor in the School of Electronic and Information Engineering, Xi'an Jiaotong University of China. His research interests include network security, insider detection, and behavioral biometrics. He is a member of the IEEE.



Yufei Chen is currently working toward the graduate degree with the Systems Engineering Institute and SKLMS Laboratory of Xi'an Jiaotong University. His research interests include computer security and insider/intrusion detection.



Xiaohong Guan (S'89–M'93–SM'94–F'07) has been the Cheung Kong professor of systems engineering, and the dean of the School of Electronic and Information Engineering since 2008, Xian Jiaotong University of China. His research interests include scheduling of complex networked resources and network security. He is a fellow of the IEEE.



Roy A. Maxion (F'08) is a research professor in the Computer Science and Machine Learning Departments, Carnegie Mellon University. He is director of the CMU Dependable Systems Laboratory whose studies include computer security, behavioral biometrics, insider/masquerader detection, usability and keystroke forensics in addition to system reliability and information assurance. He is a fellow of the IEEE.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Privacy Leakage via De-Anonymization and Aggregation in Heterogeneous Social Networks

Huaxin Li¹, Qingrong Chen¹, Haojin Zhu¹, *Senior Member, IEEE*, Di Ma², *Member, IEEE*, Hong Wen³, *Member, IEEE*, and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract—Though representing a promising approach for personalization, targeting, and recommendation, aggregation of user profiles from multiple social networks will inevitably incur a serious privacy leakage issue. In this paper, we propose a Novel Heterogeneous De-anonymization Scheme (NHDS) aiming at de-anonymizing heterogeneous social networks. NHDS first leverages the network graph structure to significantly reduce the size of candidate set, then exploits user profile information to identify the correct mapping users with a high confidence. Performance evaluation on real-world social network datasets shows that NHDS significantly outperforms the prior schemes. Finally, we perform an empirical study on privacy leakage arising from cross-network aggregation based on four real-world social network datasets. Our findings show that 39.9 percent more information is disclosed through de-anonymization and the de-anonymized ratio is 84 percent. The detailed privacy leakage of user demographics and interests is also examined, which demonstrates the practicality of the identified privacy leakage issue.

Index Terms—Data privacy, social networks security, de-anonymization, heterogeneous social networks

1 INTRODUCTION

SOCIAL networks (online social networks, mobile social networks, vehicular social networks, etc.), or social media, have been extremely popular in current days. The latest statistics show that the number of active traditional social media users has exceeded 2.7 billion [1]. Along with overwhelming popularity of social networks, people enjoy abundant functionalities and services of a variety of social networks, including sharing status updates, posting photos, communicating with others, and making friends.

Due to the different functionalities of different social networks, a user tends to sign in multiple social networks for different purposes. According to the report conducted by Pew Research Center in 2015, 52 percent of online adults use two or more social media sites such as Facebook, Twitter, MySpace, or LinkedIn [2]. Aggregating user profiles from different social networks reveals various aspects of users. It is interesting that cross-network information represents a double-edged sword. On one hand, once the user's

multiple accounts of different social networks are identified or mapped, these accounts' profiles, preferences, and activities can be collected to benefit personalization, targeting, and recommendation. The latest research pointed out that, the ads delivered by Google, one of the major ad networks, are personalized based on both users' demographic and interest profiles[3]. On the other hand, the adversary can exploit cross-network aggregation to collect the information of various aspects of the target users, which will further incur serious privacy leakage issues through inference [4], [5], [6], [7], [8]. This problem can not only exist in traditional social networks but also exist in new emerging social networks, like vehicular social networks. For example, Twitter-mobile car is able to send and receive Twitter messages, which contain the information including drivers' status, vehicle profiles, and real-time traffic notifications; Road-Speak is a voice chatting system used by daily driving commuters or a group of people who are on a commuter bus or train [9]. These vehicular social-based applications exploit traditional online social networking services, like Facebook and Twitter, and thus are also under threat of de-anonymization attack.

In this study, we take an initial step towards investigating the following two questions: i) How can we design a practical and effective cross-network aggregation scheme for heterogeneous social networks? The proposed cross-network aggregation scheme is expected to link the target user's various accounts on different social media platforms and collect the user's profile in different aspects. ii) To what degree the cross-domain aggregations can reveal the different attributes of a user (e.g., interest, demographics).

One of the fundamental challenges of bridging the different social identities of the users on different social media is

- H. Li, Q. Chen, and H. Zhu are with Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {lihuaxin003, chenqingrong}@sjtu.edu.cn, zhu-hj@cs.sjtu.edu.cn.
- D. Ma is with the College of Engineering and Computer Science, University of Michigan-CDearborn, Dearborn, MI 48128. E-mail: dmadma@umich.edu.
- H. Wen is with the National Key Laboratory of Science and Technology on Communication, University of Electronic Science and Technology of China, Chengdu 611731, China. E-mail: sunlike@uestc.edu.cn.
- X. (Sherman) Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.

Manuscript received 13 Mar. 2017; revised 16 July 2017; accepted 4 Sept. 2017. Date of publication 20 Sept. 2017; date of current version 18 Mar. 2020. (Corresponding author: Haojin Zhu.)
Digital Object Identifier no. 10.1109/TDSC.2017.2754249

that the users tend to use varying usernames (screen names) or have unequal profiles (e.g., fields such as homepage, birthday, etc.) due to the increasing privacy concerns. The process of identifying user from a social network (e.g., anonymized network) based on another social network (e.g., auxiliary network) is called ‘de-anonymization’. Recently, there is an increasing interest to study how to ‘de-anonymize’ or ‘re-identify’ users across social networks, which mainly falls to the following two categories: profile based de-anonymization and structured based de-anonymization, which either suffer from high false positive or assume the social networks are aligned.

In this study, to answer the above questions, we first present a Novel Heterogeneous De-anonymization Scheme for heterogeneous social networks, which is coined as NHDS. Different from any previous works which either focus on profile based or structure based approach, NHDS aims to integrate the merits of two kinds of approaches. The motivation is that a real-world attacker is able to leverage as much information as she can to help de-anonymize in practice. Since both user profiles and network graph topology can be collected through web crawlers, platforms’ APIs, or public datasets, a novel approach that leverages the merits of these two strategies is expected to achieve a higher performance. In particular, it first leverages the social network structure to significantly reduce the size of node candidate set. Then, it exploits user profile matching to further identify the correct mapping nodes with a high confidence. The seed nodes that act as the anchor points to align two or more heterogeneous social networks will be identified automatically.

To further investigate the privacy leakage caused by the cross-network aggregation, we apply the proposed NHDS algorithm to a large dataset involving four real-world heterogeneous online social networks, i.e., Livejournal, Flickr, Last.fm, and Myspace. We perform the de-anonymization algorithm and measure the privacy leakage arising from cross-network aggregation. The results are quite surprising in that, with the proposed de-anonymization algorithm, cross-network aggregations can reveal 39.9 percent uncovered attributes of users (e.g., interest, demographics).

To the best of our knowledge, the proposed work is the first empirical study which evaluates the impact of cross-network de-anonymization and aggregation on privacy leaking on the real-world datasets. From the privacy protection perspective, our study also reveals the potential risks to the community about user de-anonymization and information aggregation, and calls for the following research efforts on privacy-preserving personal recommendation. The major contributions of this paper can be summarized as follows:

- We propose the Novel Heterogeneous De-anonymization Scheme to de-anonymize users across heterogeneous social networks. The proposed scheme jointly exploits publicly available network structure information and user profile, which is expected to significantly increase the detection accuracy.
- We conduct extensive experiments on real-world heterogeneous social network datasets to demonstrate the effectiveness of our proposed scheme. The comparative results show that NHDS achieves high

detection accuracy and maintains a considerable recall compared with the baseline.

- To understand the consequence of real-world de-anonymization attack or cross-domain aggregation, we investigate and quantify the information leakage through network aggregation based on de-anonymized social networks. The results show that 39.9 percent information is disclosed and the de-anonymized ratio (defined in Section. 6.3) is 84 percent, which raises a serious privacy concern.

The rest of paper is organized as follows. Section 2 discusses the related research works. Section 3 models preliminary concepts and formulates the problem. The proposed approach is presented in Section 4. Then, Section 5 evaluates the results based on a set of real-world social networks. Section 6 investigates consequent privacy leakage via de-anonymization, and Section 7 concludes this paper.

2 RELATED WORK

2.1 Structure Based De-Anonymization

De-anonymizing social networks is a hot research topic in recent years. Structure based de-anonymization works are based on the assumption that the different social networks of the same group users should show the similar network topology, which can be exploited for user identification [10], [11], [12]. The observation of this kind of approaches is that a user tends to build connections with similar users they are interested in or acquainted with in different social networks. Narayanan and Shmatikov performed the de-anonymization attack to large-scale directed social networks. They designed a de-anonymization algorithm by identifying some seeds and propagating based on structure similarity [13]. In [14], Nilizadeh et al. extended Narayanan and Shmatikov’s attack by proposing a community-enhanced de-anonymizing scheme of social networks. Then, Lai [15] proposed to detect communities in social networks via user’s interests and de-anonymize users in communities. Ji et al. also designed an Adaptive De-Anonymization framework for the scenario that the anonymized and auxiliary graphs have partial overlap [16]. Some papers modeled mobility traces as graphs and presented different attacks for de-anonymizing using online social networks as side channel [17], [18]. However, in heterogeneous social networks, this assumption may not always hold due to the fact that the users of different social networks may not always be overlapping. The diversity of usage patterns on different social networks will further render the inconsistency of the network structures of the different social networks. In our proposed method, we also exploit semantic publicly available information, such as user profile, to help de-anonymize users.

Besides, Ji et al. [19], [20] conducted the comprehensive quantification on the de-anonymizability of 24 real-world social networks with seed information in general scenarios. Later, in [21], a uniform and open-source secure graph data sharing/publishing system was proposed. Li et al. proposed a graph-based framework for privacy preserving data publish, which is a systematic abstraction of existing anonymity approaches and privacy criteria [22]. Qian et al. leveraged background knowledge graph to improve the de-anonymization performance [23]. But this work mainly

focuses on de-anonymizing a graph anonymized from original graph and inferring some private attributes. Fu et al. proposed a graph node similarity measurement in consideration with both graph structure and descriptive information, and a deanonymization algorithm based on the measurement [24]. Zhang et al. targeted Twitter users in a metropolitan area by exploiting the strong geographic locality within communications on Twitter [47]. In our work, we try to de-anonymize heterogeneous social networks by considering both semantic information and structure information, and evaluate the privacy leakage after de-anonymization.

2.2 Profile Based User Matching

Public information and semantic information on social media or social network sites provide the evidence to match users of different social networks. Iofciu et al. used tags to identify users across social tagging systems such as Delicious, StumbleUpon and Flickr [25]. Olga et al. extracted features and developed supervised machine learning models which can perform entity matching between two profiles for a user by similar name and de-anonymizing a user's identity. [26] Goga et al. identified accounts on different social network sites that all belong to the same user by exploiting only innocuous activity, such as location profiles, timing profiles, language profiles, that inherently comes with posted content [27]. Vosecky et al. identified users between Facebook and StudiVZ by exploiting various profile attributes [28]. Zafarani et al. [29] conducted an in-depth investigation of this problem by defining sophisticated features to model the behavior patterns of users in selecting usernames. Korayem et al. extracted four kinds of features, i.e., temporal activity similarity features, text similarity features, geographic similarity features, social connection similarity features, and apply machine learning techniques to find correct mapping [30]. Wondracek et al. introduced a technique that narrows down user identity by examining social-network group membership stolen from browsing history [31]. Zhang et al. [32] connected social networks users by considering both local and global consistency among multiple networks, but they treat both two consistencies as features and train an energy-based learning model. [33] provided a preliminary evaluation on the combination of graph structure information and profile information for de-anonymizing, but didn't evaluate how much information will be leaked through the de-anonymization. In [34] and [35], the first privacy-preserving personal profile matching scheme for mobile social networks was proposed by Li et al. In this scheme, an initiating user can be identified from a group of users the one whose profile best matches with his/her, with limited risk of privacy exposure. Later, two novel fine-grained private profile matching protocols were designed in [36], [37]. Different from these works, our proposed approach uses social structure to narrow down the candidate sets in order to achieve higher accuracy.

3 MOTIVATION AND MODELING

3.1 Motivation

As introduced above, the Novel Heterogeneous De-anonymization Scheme (NHDS) integrates both network

structure and public information to de-anonymize social network users. On one hand, network structure and topology can be leveraged to de-anonymize social networks users with considerable accuracy as reported in state-of-the-art [13], [14]. However, this kind of approaches requires enough overlapping to ensure the accuracy by revisiting nodes and correcting early false mappings [13], thus can be less effective when the two networks are heterogeneous and not well aligned. On the other hand, profile information is useful to identify a person's different social networks accounts [28], [30], [47], but it may also cause many false positives in a large scale de-anonymization due to the same or similar profile attributes of different persons (e.g., same or similar nicknames of different persons). Since a social network contains a huge number of users, the possibility that different persons have the same or similar online profiles can be non-negligible.

The first step of NHDS is that social network structure is leveraged to significantly reduce the size of node candidate sets. Then, user profile information is exploited to further identify the correct mapping nodes. Our motivation of this design is based on the observation that the graph structures (e.g., community, neighborhood) of social networks are likely to be similar for the same user groups [14], [15]. So community and neighborhood are employed to generate the candidates set and guide mapping process in a general view. Then in a specific view, a person is likely to have very similar profiles and generate same evidence on different social networks. Thus public available profile information is used to decide node mappings with a high confidence. Before presenting the scheme, we formulate the graph model, profile information model, and attack model in remaining part of this section.

3.2 Graph Structure Modeling

Social network structure is usually represented as a graph, where each user is a node in the graph, and connections between a pair of users are represented as edges. Let $G = (V, E)$ represent a social network graph where V is a set of users and $E \subseteq V \times V$ is a set of directed/undirected links between users. $e(v_1, v_2)$ means that v_1 and v_2 are in friend relationship or follower/followee relationship where $e \in E, v_1, v_2 \in V$. As the important structures in the social network graph, *neighbor* and *community* are formally defined as follows:

Definition 1. *A neighbor set R of a user $v_i \in V$ is a set of users $R_i = [v_j]_{j=1, \dots, \forall v_j : \exists e(v_j, v_i)}$, who are directly in friend relationship or follow relationship with v_i . We further define a function α to form the neighbor set R_i of v_i , i.e., $\alpha(v_i) = R_i$.*

Definition 2. *A community C in a social network graph is a disjoint partition of vertices in $G(V, E)$. Formally, we denote communities in a graph as $C = \{C_1, C_2, \dots, C_k\}$, where $C_i \neq \emptyset$ and $C_i \cap C_j = \emptyset$ if $i \neq j$ for $1 \leq i, j \leq k$. $\forall C_i \in C, V_{C_i} \subset V$ and $E_{C_i} \subset V_{C_i} \times V_{C_i}$. In this paper, communities are defined by Infomap algorithm [38], which uses the probability flow of random walks on a network and decomposes the network into modules by compressing a description of the probability flow [38].*

3.3 Profile Information Modeling

As platforms aiming at attracting attention, boosting self-presentation, promoting and sustaining social capital, social networks must allow part of user profile information to be available to public. The amount of publicly available profile

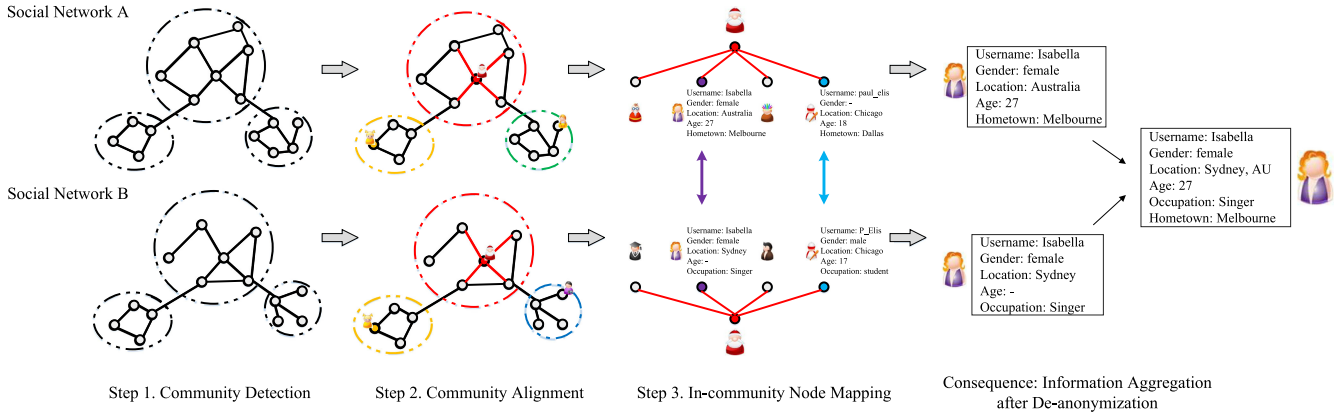


Fig. 1. Overview of our scheme.

information on social networks varies from each other, according to the platform defined and/or user defined privacy settings. For instance, Twitter allows users to follow other users without permissions and most profile are in public, while on Facebook, one user needs to send request to another for becoming ‘friends’, then the profile becomes visible. Meanwhile, a Facebook user might/might not choose to show his/her gender, status, hometown, on this platform. To exploit profile information across heterogeneous social networks, we first give a uniform definition:

Definition 3. Let $X_i = [x_{ik}]_{k=1\dots d}$ denote a set of attributes associated with the user $v_i \in V$ (for instance, username (screen name), location, self-description, etc), where d is the number of types of attributes and x_{ik} records the content of the k th attribute of user v_i . If a user v_i 's j th attribute is not available on the social network (e.g., a user chooses not to show her hometown on Twitter), $x_{ij} = \text{null}$. Note that a user may have several vectors modeling different profiles in social networks where he/she has an account. Common attributes between two vectors will be used to compute the profile similarity.

Since heterogeneous social network platforms contain different kinds of profile information, and some of them contains semantic or syntactic meaning, mapping two users' accounts from two heterogeneous social networks is similar to an ontology matching problem. In general, ontology matching determines an alignment for a pair of ontologies O_1 and O_2 . Each ontology consists of a set of discrete attributes which are usually represented in the form of tables, classes, properties, and determines as output the relations. In our problem, profile matching can be defined as follows:

Definition 4. Given two profiles, $p_A = \{x_1, \dots, x_A\}$ and $p_U = \{x_1, \dots, x_U\}$. If $\text{type}(x_i) = \text{type}(x_j)$ for two attributes $x_i \in p_A$ and $x_j \in p_U$, the similarity between the two attributes is defined as

$$\text{sim}_a = \text{matchScore}(x_i, x_j). \quad (1)$$

Here what the similarity is depends on which attribute is considered, it can be value similarity for ages or genders, string similarity for screen names, text similarity for descriptions or tweets, semantics similarity for locations or hometowns, etc., as presented in Section 4.4. Then the similarity of profiles is computed by

$$\text{sim}_p = \frac{\sum_{r=1}^t w_r (\text{sim}_a)_r}{t}, \quad (2)$$

where w_r is the weight given to attributes, and t is the number of attribute pairs of the same type between two profiles.

In Section. 4.4, we will show how to match users according to various profile attributes.

3.4 Attack Model

We assume two heterogeneous social network graphs G_A and G_U . G_A is denoted as the anonymous network graph and G_U is the auxiliary network graph. Note that the graph here is not necessary to be the whole graph of a social network, the de-anonymization attack can be conducted on partial graphs (i.e., subgraphs) collected by the attacker. That is to say, the attacker is able to obtain a subgraph $G = (V, E)$ and profile attributes X_i corresponding to $v_i \in V$ through published datasets or crawling sites. The goal of the attacker is to learn more information of the users across different networks by mapping users in G_A to users in G_U . To achieve this goal, the attacker needs to identify user accounts from two different social networks that belong to a same person in large scale and with a high confidence. This problem can be formally defined as follows.

Problem 1. Given (1) two different social network graphs $G_A = (V_A, E_A)$ and $G_U = (V_U, E_U)$, (2) sets of attributes X_i and X_j of $v_i \in V_A$ and $v_j \in V_U$ respectively, finding user mappings $v_i \leftrightarrow v_j, v_i \in V_A, v_j \in V_U$ that belong to the same real persons accurately by iteratively computing

$$\underset{v_i \in \text{Cand}_A, v_j \in \text{Cand}_B}{\text{argmax}} S(X_i, X_j), \quad (3)$$

where S is a function to compute similarity between X_i and X_j , as shown in Equation (2). Cand_A and Cand_U are two candidate sets for potential correct mappings generated by community and neighbor structure in G_A and G_U , respectively.

4 NOVEL HETEROGENEOUS DE-ANONYMIZATION SCHEME

In this section, we introduce our proposed Novel Heterogeneous De-anonymization Scheme.

4.1 Scheme Overview

Fig. 1 illustrates our proposed scheme which has three main steps: (1) Communities Detection: communities in both networks are detected according to graph structure, (2)

Communities Alignment: seeds are automatically identified based on profiles, and communities that contain the same pairs of seeds are aligned, (3) In-community node mapping: in each pair of aligned communities, nodes with high similarity score, which is computed by profile similarity in Equation (2), is accepted as a mapping, and mapping process is propagated to the neighbors. Algorithm. 1 presents the whole procedure, and the details and time complexity are introduced in the following sub-sections.

Algorithm 1. Algorithm of Proposed Scheme

Input: $G_A < V_A, E_A >, G_U < V_U, E_U >$, threshold θ

Output: Mappings of users μ'

//Communities detection

$C_A = \text{Infomap}(G_A)$

$C_U = \text{Infomap}(G_U)$

//Communities alignment

$\mu = \text{SelectSeeds}(V_A, V_U)$

$\text{CommPairs} = \text{AlignCommunities}(C_A, C_U, \mu)$

//In-community node mapping

$\mu' = \text{InCommunityMapping}(\text{CommPairs}, \mu, \theta)$

return μ'

4.2 Communities Detection

The goal of first step is to partition social network graphs G_A and G_U into two sets of communities $C_A = \{c_1, \dots, c_m\}$ and $C_B = \{c_1, \dots, c_n\}$. We devise the communities detection algorithm based on Infomap algorithm [38], which has a low time complexity, to partition disjoint, non-overlapping communities C_A and C_U for two graphs, respectively. In brief, Infomap finds the shortest multilevel description of the random walker therefore giving us the best hierarchical clustering of the network—the optimal number of levels and modular partition at each level—with respect to the dynamics on the network. So another merit of using Infomap algorithm is that it generates C_A, C_U with different scales at different levels so that we can choose communities with similar scale for aligning. The algorithm for communities detection and division is denoted as the $\text{Infomap}(\cdot)$ function in Algorithm. 1, and the time complexity is $O(|E|)$.

4.3 Communities Alignment

For aligning communities $C_i \in C_A$ and $C_j \in C_U$, [14] proposes to treat each community as a node in a graph, then propagate the communities mapping process from some ‘community seeds’ using an improved version of [13]. However, in practice, we find that communities can be more easily aligned given the publicly available profile information. We choose the username (or screen name) to identify seeds for two reasons. First, the username (screen name) must be available on every social network’s website, so the attacker has enough chances to obtain or crawl them. Second, as shown in [13], [30], the possibility that two accounts with the same usernames (screen names) do not belong to a user is less than 5 percent. Thus we design the following algorithm, which aligns C_A and C_U according to the number of same usernames in communities according to the algorithm described as the following two steps.

- 1) The first step is to find all user pairs with same usernames $\mu = \{\dots, (u_i, u_j)_k, \dots\}$ where $u_i \in V_A$ and

$u_j \in V_U$. Greedy searching will cause a high complexity of $O(|V_A||V_U|)$. Instead, this process can be implemented by a hash table so that the time complexity can be reduced to $O(|V_A| + |V_U|)$. This procedure is denoted as $\text{SelectSeeds}(\cdot)$ function in Algorithm. 1.

- 2) In the second step, an initial confidence score $cs_{i,j}$ (that indicates whether two communities should be aligned) for each pair of communities (C_i, C_j) , where $C_i \in C_A, 1 \leq i \leq m, C_j \in C_U, 1 \leq j \leq n$, is set as 0. For each pair $(u_p, u_q) \in \mu$, $cs_{i,j}$ is added by one, given $u_p \in C_i$ and $u_q \in C_j$. Then, all confidence scores cs for communities pairs are examined, if $cs_{i,j}$ exceeds a threshold θ_{cs} , C_i and C_j are aligned. The time complexity of this step is $O(|\mu|) < O(|V_A| + |V_U|)$. This procedure is denoted as $\text{AlignCommunities}(\cdot)$ function in Algorithm. 1.

The overall complexity of communities alignment algorithm is $O(|V_A| + |V_U|)$, as described above. Our overall evaluations show that our communities division and alignment only slightly reduce the recall rate.

4.4 In-Community Node Mapping

Algorithm. 2 describes our in-community node mapping algorithm. Within each pair of aligned communities, a propagating and mapping algorithm (in $\text{Propagation}(\cdot)$) is performed locally. This algorithm takes two graphs of communities $G_{c1} = (V_{c1}, E_{c1}), G_{c2} = (V_{c2}, E_{c2})$ and the set of seeds in these communities $\mu_{c1,c2}$ selected in the previous step as input. It iteratively finds new mappings in the neighbor sets of seeds in $\mu_{c1,c2}$, and extends mapping process based on graph structure. At each iteration, the algorithm computes similarity scores within two neighbor sets $R_u = \alpha(u)$ and $R_v = \alpha(v)$, which are generated by two already mapped users u and v . It picks a user r_u in R_u and computes similarity score with users in R_v and find out a r_v with the highest score. The similarity score is computed by the $\text{MatchScore}(\cdot)$ function in Algorithm. 2, which will be discussed later. If the score exceeds a pre-defined threshold θ , r_u and r_v are accepted as a successful mapping. If an already mapped node is mapped to another node with a higher similarity score, the previous mapping is replaced by the new mapping with higher score. The process is halted if no new mapping is explored, and unvisited nodes in propagation process of both communities are gathered to compare to find remaining mappings. The time complexity of propagation is $O(\min\{|V_{c1}|, |V_{c2}|\}d_{c1}d_{c2})$, where d_{c1} and d_{c2} are bounds on the degree of the nodes in V_{c1} and V_{c2} , respectively.

To compute the similarity score between two users (nodes), profile matching are exploited. Due to the variety of profile attributes, $\text{MatchScore}(\cdot)$ implements an ‘‘if-then’’ rule to compute similarity scores of different kinds of profile attributes in different methods. After that, an overall similarity score of the two users is given by assigning weights empirically to score of different attributes as the form of Equation (2). Different techniques are leveraged to compute similarity score of different kinds of attributes.

4.4.1 Value Matching

Direct value matching can be used to match non-string literal attributes, such as gender, birth date, and personal

website (or link). This kind of attributes usually have fixed formats on social networks, which indicates high confidence of rejecting (if genders of two users are different) or accepting (if birth date or personal websites of two users are same) a potential mapping.

Algorithm 2. Algorithm of the InCommunityMapping(\cdot)

Input: community pairs $CommPairs$, seeds μ , threshold θ

Output: μ' with more mappings of users

for $(C_a, C_u)_j \in CommPairs$ **do**

$\mu_j = Propagation(C_a, C_u)$

end

return $\mu' = \bigcup_{j=1, \dots, len(CommPairs)} \mu_j$

Procedure Propagation R_1, R_2

$\mu_j \subset \mu$ // the seeds set of $(C_a, C_u)_j$

while exists $\langle v_1, v_2 \rangle \in \mu_j$ is *unvisited* **do**

$R_1 = \alpha(v_1), R_2 = \alpha(v_2)$

for r_1 in R_1 **do**

for r_i in R_2 **do**

scores[r_1].add(MatchScore(r_1, r_i))

end

if MAX(scores[r_1]) $> \theta$ **then**

r_{max} = user with MAX(scores[r_1])

add $\langle r_1, r_{max} \rangle$ into μ_j and mark *unvisited*

end

end

Mark $\langle v_1, v_2 \rangle$ *visited*

end

return μ_j

4.4.2 Syntactic Matching

Syntactic matching is applied to attributes that are usually shown as strings (e.g., username and person name). These attributes on different social networks often have editing differences, such as difference among “Jones, David”, “David Jones”, and “D. Jones”. So string matching metric can be used for syntactically matching these attributes. In order to avoid the influence of abbreviation or acronym, Monge-Elkan algorithm, a recursive string matching algorithm, is applied [39]. The basic idea of this method is to break input string into tokens. Then the best matching token are compared to get the score as follows.

$$MongeElkan(A, B) = \frac{1}{|A|} \sum_{|A|}^{i=1} \max\{dist(A_i, B_j)\}_{j=1}^{|B|}, \quad (4)$$

where A and B are two strings, and $dist()$ refers to a secondary distance function used to compute similarity between tokens of A and B . In a lot of functions computing edit-distance, *Jaro-Winkler similarity* is chosen as the secondary distance function in our problem, due to its remarkable performance in previous research on name-matching tasks [40]. Monge-Elkan algorithm returns 1 if two string are fully matched or one abbreviates the other; return 0 is there is no match between the two strings.

4.4.3 Keywords Matching

Keywords matching is a tool to handle attributes that are in the form of texts (e.g., self-description of users or ‘aboutme’). Since the contents a person posts and the words

the person uses are likely to be similar, the similarity of texts, which is usually measured by some keyword-based matching methods, are used to determine whether two texts come from a same person. Given two texts, we first compute TF-IDF weights, which normalize each word count by the number of people that used it (in the comparison set, say, neighbour set R mentioned above), to get two weights vectors d and e , respectively. Then we compute the cosine similarity between d and e as the text similarity score

$$Cosine(d, e) = \frac{d \cdot e}{\|d\| \|e\|} = \frac{\sum_{i=1} d_i e_i}{\sqrt{\sum_{i=1} d_i^2} \sqrt{\sum_{i=1} e_i^2}}. \quad (5)$$

4.4.4 Semantic Matching

Previous three matching methodologies are based on similarity of string or text. However, some of attributes, such as locations and hometown, might be semantically equal when the term or words are totally different. For example, location attribute on Flickr is referred as the country where a user is, where the location attribute on Myspace can be filled in freely by the user. So a user who lives in Michigan State of America might have location of “America” on Flickr and “Michigan” on Myspace. If we only consider string similarity, “America” and “Michigan” have a large edit distance, and might be considered to be from different users literally. But they are semantically related and possible to be from the same user. In order to solve this problem, we use GeoNames database [41] to check whether two locations are semantically related. If they are actually a same location, we give a high score; if they are in inclusion relationship (say, a city in a state or a state in a country), a medium score is given; if they are two independent locations, a low score is assigned.

With these four methods for computing similarity of attributes, $MatchScore(\cdot)$ is able to compute the similarity score of users, and return it to $Propagation(\cdot)$.

5 EVALUATIONS OF PROPOSED SCHEME

In this section, we evaluate our proposed NHDS scheme by conducting experiments on a set of real-world social networks data.

5.1 Datasets

The datasets of four real-world heterogeneous online social networks, i.e., Livejournal, Flickr, Last.fm, and Myspace, are obtained from [32]. The datasets include node information, edge information, and profile information of a subset of users of these social networks.

- *LiveJournal* is a social networking site and blogging platform that allows users to find each other through journaling and interest-based communities. The dataset consists of 3,017,286 users and 19,360,690 friend relationship.
- *Flickr* is an image hosting online community for sharing, storing, and organizing photos. The dataset consists of 215,495 users and 9,114,557 friend relationship.
- *Last.fm* is the world’s largest online music catalogue and has been recognized as a popular social network for music enthusiasts. Last.fm builds detailed profiles of users’ musical tastes and preferences. The

TABLE 1
Statistics of Social Networks

Network	Nodes	Edges	Av. Degree
Livejournal	3,017,286	19,360,690	12.83
Flickr	215,495	9,114,557	85.59
Last.fm	136,420	1,685,524	24.71
Myspace	854,498	6,489,736	15.19

dataset consists of 136,420 users and 1,685,524 friend relationship.

- *MySpace* is a social networking website offering an interactive, user-submitted network of friends, personal profiles, blogs, groups, photos, music, and videos. The dataset consists of 854,498 individuals and 6,489,736 friend relationship.

We build undirected social network graphs according to ‘friend’ or ‘follow’ relationship in these social networks. The statistics of the graphs are shown in Table. 1. These social networks not only provide different services and have different utility, but also have different graph properties. For example, Flickr has an average degree of 85.59 while the average degree of Livejournal is only 15.19. The heterogeneous structure increases the difficulty of de-anonymization.

In order to evaluate the results, we obtain the ground truth data from [32], [42], which contain pair-wise matched user id of two social networks. The data were originally collected by Perito et al. [42] through Google Profiles service by allowing users to integrate different social network services.

5.2 Experiments

Since our scheme is a combination of graph structure and publicly available profile information, we evaluate the results by comparing our approach with approaches that only exploit profile information, and approaches that are only based on graph structure, respectively. To quantitatively evaluate the algorithm, we consider the two widely-used metrics:

- *Precision*: In all mappings returned by the de-anonymization algorithm, the percentage of correct mapping. Since our goal is to find out correct mappings rather than find out incorrect mappings, the concept of precision here is same as the concept of accuracy.
- *Recall*: The percentage of correct mapping retrieved by algorithm in all mappings in ground truth.

The codes of experiments are written in Python and the programs are ran on a server with Intel Xeon 2.4 GHz 14-core CPU and 64 GB memory.

5.2.1 Evaluations of NHDS

As introduced above, the θ_{cs} and θ are the only two setting parameters of our scheme, so we set $\theta_{cs} = 1$ and vary θ to evaluate the results in the following parts. Due to the users’ awareness of privacy protection and social networks’ privacy settings, few common profile attributes are available for all users across multiple social network graphs. In order to evaluate our proposed NHDS scheme, we select username (screen name), which is widely available for all users of all social networks in our datasets, as profile information, and perform overall evaluations first. Then, comparative experiments by considering limited graph structures and more profile attributes are conducted on part of users to show that the performance under different situation. The direct profile-based matching represented by [28], i.e., computing profile similarity between each user in one social network and all users in the other social network and find the most similar one, is used as the baseline.

Fig. 2 shows how our NHDS scheme outperforms the profile-based matching by tuning threshold θ . The precision of our approach is obviously higher than the baseline by slightly sacrificing the recall. It reflects that graph structures (community/neighborhood) are useful to filter incorrect matchings, thus increasing the precision. As introduced in Section 4.4, the threshold θ is a similarity criterion that accepts a pair of nodes as a mapping in our algorithm, i.e., if the similarity score between two nodes exceeds the θ , the two nodes are accepted as a potential mapping. So, the higher θ is set, the more similar the accepted nodes are, and thus fewer potential mappings will be returned. So θ actually reflects the trade-off between precision and recall. An attacker can choose θ according to his/her requirement of this trade-off in practice. When the threshold is set to 0.9, the precision of matching users can be more than 90 percent with a recall of 40 percent for Flickr-Lastfm and Livejournal-Lastfm. The results show that large scale accurate de-anonymization can be performed.

Impacts of Graph Structure. Sometimes it is difficult to access a full view of a social network graph due to access limitations and privacy policies. Especially from the attacker’s perspective, data that the attacker can collect are usually processed data where proper noise has been added. So it is interesting to investigate whether the attack is still feasible when limiting an attacker to a restricted view on the graph. We simulate different portions of graph to be analyzed by randomly removing different percentages (i.e., 10, 20, 30, 40, and 50 percent) of edges from original data. As shown in Fig. 3, removing certain portions (e.g., 10-40 percent) of

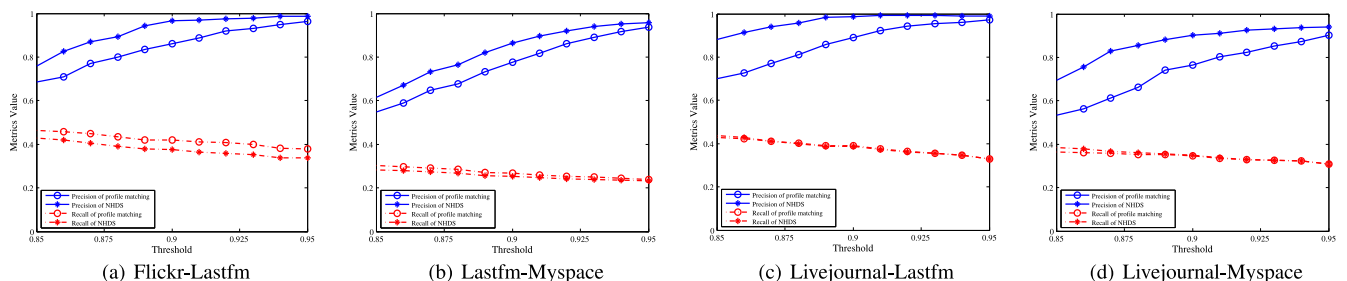


Fig. 2. Performance comparisons between profile-based matching and proposed NHDS.

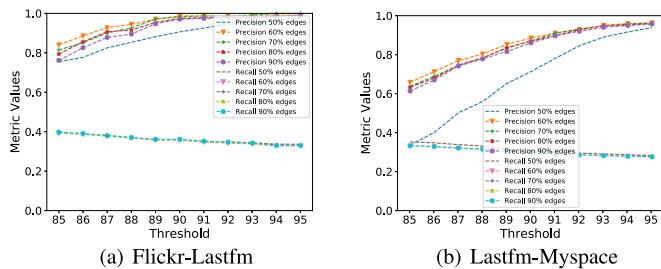


Fig. 3. Performance after randomly removing edges. In the legend, for instances, “90 percent edges” means 90 percent edges are remaining in the graph, which also means 10 percent edges are removed.

edges will not obviously decrease the precision. After removing 50 percent of edges, precision noticeably drops because the graph structure is significantly affected and cannot act as the guide to reduce false positive. Meanwhile, the recall is almost not affected. These results show that, though randomly removing of edges disturbs the graph structure, the attacker is still likely to leverage profile attributes to precisely identify correct mappings. The comparison between graph based approach and our approach in Section 5.2.2 also shows involving profile attributes can help identify mappings when graph structure is not well aligned.

Impacts of Multiple Profile Attributes. Then, we consider more profile attributes, including username, nickname, status, gender, location, and aboutme, to de-anonymize users between Flickr and Myspace, because the two social networks have the most available common profile attributes. We set the weights as $w(\text{username}) = 0.2, w(\text{nickname}) = 0.2, w(\text{status}) = 0.1, w(\text{location}) = 0.2, w(\text{gender}) = 0.2, w(\text{aboutme}) = 0.1$ and compute similarity scores as proposed in Section 4.4. The performance is compared with the scenario that only username is used to calculate similarity scores. Fig. 4 shows that when more profile attributes are considered between Flickr and Myspace, the precision is improved when the recall is kept the same. Similar conclusions can also be drawn on other social networks. For instances, 46 percent precision by considering name, location, links, aboutme versus 40 percent precision by considering username only when recall are both 29 percent for de-anonymization between Livejournal and Lastfm.

5.2.2 Comparisons with Profile-Based Approaches

Previous studies exploit various user profile information to connect individuals between social networks, including usernames [28], tags [25], activities [27], group membership [31], and multiple kinds of profile attributes [29], [30], [32], [44], [46]. In order to reflect properties of our approach, NHDS is further compared with some existing profile-based approaches, including direct profile matching [28], MNA [44], SiGMa [46], and COSNET [32]. Fig. 6 shows the precision and recall of these approaches for instances. We can clearly observe that the precision and the recall is always a trade-off. This is because, if targeting at a higher precision, an algorithm must have strict criteria of identifying a mapping to ensure its correctness, thus resulting in lower recall; if targeting at a higher recall, i.e., more mappings are returned, the algorithm has to loose criteria to accept more possible mappings, as well as more potential false positive. Compared with other algorithms, NHDS

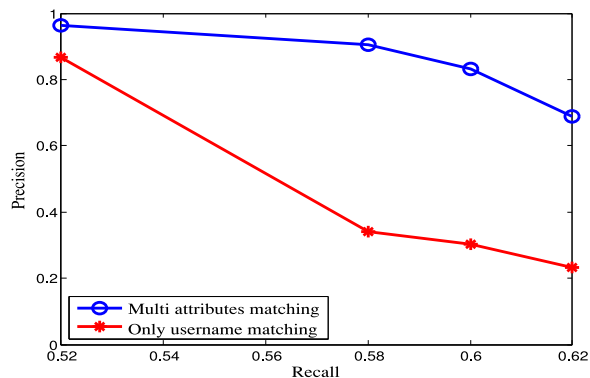


Fig. 4. Comparison of multi-profiles and username matching.

provides the highest precision though sacrificing the recall. We believe that given a low precision, the high recall is nonsense—the attacker still cannot be confident about whether obtained mappings are correct. So our approach has its benefits for providing the most accurate result.

5.2.3 Comparisons with Graph-Based Approaches

As mentioned above, numerous graph-based de-anonymizing algorithms have been proposed. However, only a few of them are suitable to real-world heterogeneous social networks for various reasons. Some techniques are constrained by their restrict requirements of social networks of the same size (or same number of nodes) [10], [12], sybil users [43] or high computation capability for large scale networks [17], while others have only been evaluated between noisy graph and its original graph [14], [16], [23]. For reference, we test the well-known graph-based NS algorithm proposed by Narayanan and Shmatikov [13] and percolation-based de-anonymization algorithm [44] using the open-source evaluation system proposed in [21]. As a result, only few correct mappings are reported by the two previous algorithms on the heterogeneous social network datasets feeding more than 100 seeds. One possible reason of the unsatisfactory performance is pure graph-based approaches require enough overlaps of the network graphs to propagate and correct false mappings at the beginning of the mapping [13]. But it is usually difficult to obtain datasets with ideal overlaps from two heterogeneous social networks, which limits the performance of graph-based approaches in practice. According to [13], 30.8 percent of the mappings were re-identified correctly between a Twitter dataset (Av. degree of 37.7) and a Flickr dataset (Av. degree of 32.2), which is far away from our results on more heterogeneous networks datasets. The results show that introducing profile attributes of nodes obviously increases the successful rate of de-anonymizing.

6 REAL-WORLD PRIVACY LEAKAGE EVALUATION

We have shown that our approach is able to de-anonymize users accurately in a large scale. As shown in the last step in Fig. 1, aggregated profiles from de-anonymization reveal more privacy of the user. To understand the severity of de-anonymization attack and provide reference of social network privacy protection, it is worth to investigate to what extent the user’s privacy will be exposed. To quantify the user privacy leakage after de-anonymizing, we retrieve de-anonymized users returned by our approach as described

(a) Flickr's profile setting page

(b) Lastfm's profile setting page

Fig. 5. An example to illustrate platform preserved information.

in Section. 5.2.1 ($\theta = 0.9$), and quantify real-world information leakage through de-anonymization.

6.1 Profile Appearance on Different Social Networks

Through the profile aggregation of accounts from different social networks, more previously unknown information of the same person can be obtained. For the ease of presentation, we first introduce and define two types of the previously unknown information which can be gained after aggregation as follows.

6.1.1 Platform Preserved Information

Different social networks contain different profile information, which can be exposed in public, according to the platform settings and utility. Fig. 5 illustrates an example of user profile setting pages of Flickr and Lastfm, where users can choose to fill in this page to demonstrate their profiles to the public. We can see that the Flickr allows users to show more profile information, including the singleness, occupation, hometown, current city, than Lastfm, as highlighted in red in Fig. 5a. As a result, these profile information can be exposed through Flickr, while being preserved by Lastfm.

Similarly, Table. 2 lists profile attributes revealed on social networks contained in our datasets. For example, Flickr provides users' gender and location on the platforms, while gender and occupation of users are available in Myspace. Also for the location information in Flickr and occupation information in Myspace, only one of the two platforms contains this information, while the other one preserves it. So, we denote this kind of information as *platform preserved information*. Given pairs of users de-anonymized between two social networks, the attackers might know more kinds of profile attributes of users, say the location information of Myspace

users or occupation information of Flickr users. The aggregations of platform preserved information is coined as *platform preserved information aggregation*, which aggregate attributes that one platform is preserved through de-anonymizing from another platform.

6.1.2 User Preserved Information

On the other hand, though two social networks contain common profile information settings (e.g., the gender information appeared on Flickr and Myspace in our example), some profile information can still be preserved from the public by the users. That's because a person may choose not to fill in and show all his/her profile information online. Fig. 7 shows online profiles of two real-world Flickr users (important information is blurred in order to preserve users' privacy). Compared with the user B, the user A chooses to show his/her contact email, hometown, and current city to public while hides his/her occupation. We denote this kind of information as *user preserved information*. The attackers may have the chances to uncover information that the users preserved on a social network through de-anonymizing and aggregation. For example, the attacker can still collect the Flickr user A's occupation from Myspace or the user B's

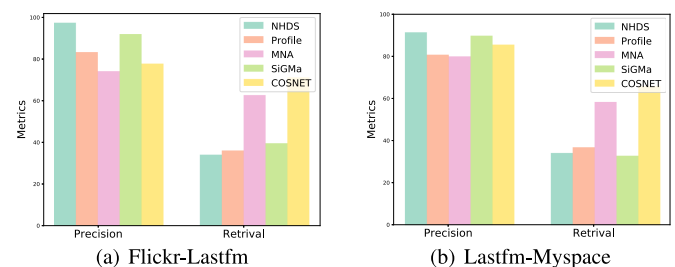


Fig. 6. Comparisons of profile based approaches.

TABLE 2
User Profile Revealed on Social Networks

	gender	age	links	status	interests	location	hometown	education	occupation	aboutme	orientation, income, etc.
Livejournal			✓		✓	✓		✓		✓	
Flickr	✓		✓	✓		✓	✓		✓	✓	
Lastfm	✓	✓	✓			✓				✓	
Myspace	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓

address from other social networks. We call it as *User preserved profile uncovering*.

In summary, the platform preserved information is the information that attackers can not obtain from this social network platform but can be collected from other social networks, due to the platform settings. And the user preserved information is the information that users choose not to show on the platform though they have the options to show. We will quantify these two information leakage in the following parts.

6.2 Platform Preserved Information Aggregation

First, we evaluate the information leakage from *platform preserved profile aggregation*, i.e., aggregation of different kinds of attributes between two social networks. We compute the percentage of attributes exposure of de-anonymized users from two heterogeneous social networks. Table 3 shows the attributes that are available on one platform but not available on the other platform, due to the platform settings, and corresponding exposed ratio. The exposed ratio of an attribute is calculated by the percentage of users who show this attribute online within all the de-anonymized users. From Table 3, we can observe various personal information, e.g., orientation, income, ethnicity, are possible to be exposed from Myspace, while almost half users in Flickr reveal their locations and occupation. So the attackers are possible to obtain more complete user profiles and their locations after de-anonymizing. Similarly, a de-anonymized Livejournal user's gender and age are likely to be collected through its matched Lastfm account. So de-anonymization across heterogeneous social networks makes it possible to learn users information from more aspects, and construct the more comprehensive profiles of users.

6.3 User Preserved Information Uncovering

Then, we evaluate the information leakage of *uncovering user preserved attributes*—attributes that are available on the settings of both two social networks but users might not fill in on both sites. For the convenience of explanation, we first denote some concepts. A common attribute that is possible to be shown on two platforms

can be classified into three types according to the users' settings:

- *Known attribute* means the attribute that users show on both platforms. So the attackers have already known the attribute of the users without de-anonymization from an auxiliary network.
- *Unknown attribute* means the attribute that users did not show on both platforms. So the attackers can not obtain the attribute even though two accounts are matched.
- *De-anonymized attribute* means attribute that users show on one site but not on the other. So the attacker can collect the attribute of users through de-anonymizing.

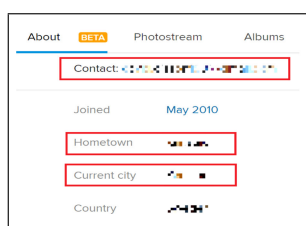
For example, as shown in Table 2, both Flickr and Myspace provide settings of gender, hometown, and occupation to users. If a user has his gender on the both platforms, shows hometown on Flickr but not on Myspace, and doesn't show occupation on both sites, gender is a *known attribute*, hometown is a *de-anonymized attribute*, and occupation is a *unknown attribute*, as defined above.

Then, in order to evaluate how many users' attributes can be obtained after de-anonymization, we further define different portions of users for a specific attribute:

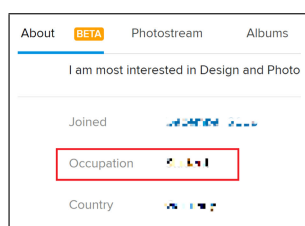
- *Known portion* means percentage of users who show the attribute on both platforms.

TABLE 3
Platform Preserved Information Leakage

De-anonymization	Source	Attributes	Exposed Ratio
Flickr	Flickr	occupation	45.52%
		location	51.36%
		age	49.89%
		interests	25.27%
		orientation	21.09%
		bodytype	15.60%
		ethnicity	17.14%
Flickr-Myspace	Myspace	religion	22.41%
		children	19.56%
		smokedrink	16.70%
		education	18.90%
		occupation	20.87%
		income	5.71%
		schools	28.13%
Livejournal	Lastfm	email	3.86%
		schools	36.48%
		interests	62.66%
		birthdate	74.24%
		age	79.16%
Livejournal-Lastfm	Lastfm	gender	100.00%



(a) Flickr profile of user A



(b) Flickr profile of user B

Fig. 7. An example to illustrate user preserved information.

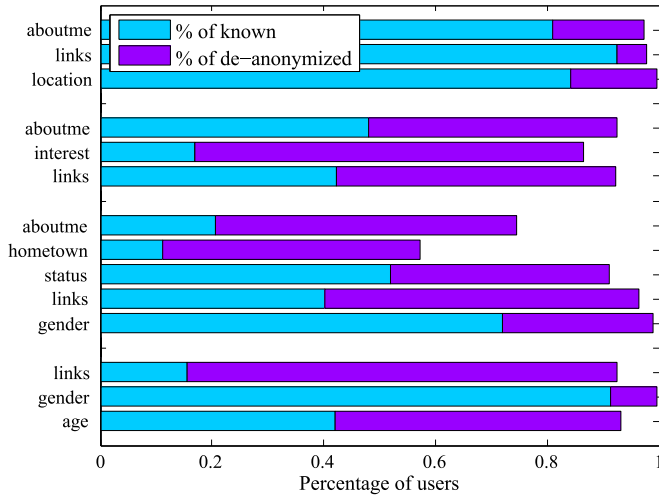


Fig. 8. Information gain through de-anonymization.

- *Unknown portion* means percentage of users who did not show the attribute on both platforms.
- *De-anonymized portion* means percentage of users who show the attribute on one platform but not on the other.

Fig. 8 shows the comparison of de-anonymized portions and known portions of attributes in different set of de-anonymization experiments (Livejournal-Lastfm, Livejournal-Myspace, Flickr-Myspace, Lastfm-Myspace from top group of bars to bottom group of bars). Some attributes, such as *links*, *hometown*, *interests*, contain high de-anonymized portions, which exceed 45 percent. And the average percentage of de-anonymized portion (Equation. (6)), which represents the information gained from de-anonymization, is 39.9 percent. It indicates notable information leakage through de-anonymization.

$$Info_Gain = \frac{de-anonymized\ portion + known\ portion}{known\ portion} - 1. \quad (6)$$

Furthermore, we evaluate how much previously invisible information the attackers can obtain from the de-anonymization attack. Since the known portion of attribute is already visible for the attacker, we define the de-anonymized ratio as

$$Radio = \frac{de-anonymized\ portion}{1 - known\ portion} = \frac{de-anonymized\ portion}{de-anonymized\ portion + unknown\ portion}. \quad (7)$$

Fig. 9 shows the de-anonymized ratio of attributes in different de-anonymization experiments (Livejournal-Lastfm, Livejournal-Myspace, Flickr-Myspace, Lastfm-Myspace from top group of bars to bottom group of bars). The average ratio is up to 0.84. The result shows that the attackers can obtain a great portion of previously unknown profile information through de-anonymization.

Our quantified evaluation shows that the privacy leakage through de-anonymization attack across real-world social networks is severe. Privacy preserving strategies and

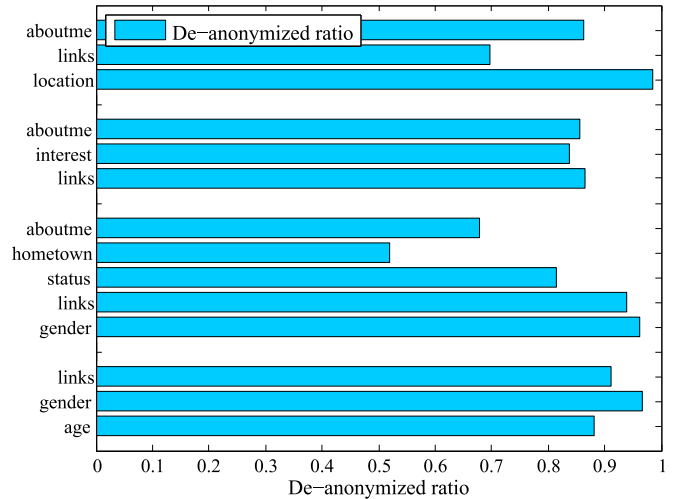


Fig. 9. De-anonymized ratio of attributes.

mechanisms for both protecting privacy of users and maintaining social network utility are still open research problem.

7 CONCLUSION

In this paper, we propose a practical Novel Heterogeneous De-anonymization Scheme for de-anonymizing real-world heterogeneous social networks, and evaluate and quantify the following privacy leakage. NHDS is a de-anonymizing scheme that exploits the network graph structure to significantly reduce the size of candidate set, and use user profile information to identify users with a high confidence. The performance evaluations of NHDS based on a dataset of four real-world social networks show that it achieves a high precision with a slight sacrifice of recall. We further quantify privacy leakage through de-anonymization. Evaluations show that notable portions of user information is disclosed. Privacy preserving in social networks is still an open challenge.

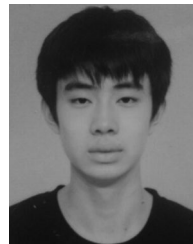
ACKNOWLEDGMENTS

This work is supported by National Science Foundation of China (no. U1401253, 61672350, U1405251, and 71671114), National High-Tech R&D (863) Program (no. 2015AA01A707).

REFERENCES

- [1] D. Chaffey, "Global social media research summary," 2016. [Online]. Available: <http://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/>
- [2] M. Duggan, N. Ellison, C. Lampe, A. Lenhart, and M. Madden, "Social media site usage 2014, pew research center," 2015. [Online]. Available: <http://www.pewinternet.org/2015/01/09/social-media-update-2014/>
- [3] W. Meng, R. Ding, S. P. Chung, S. Han, and W. Lee, "The price of free: Privacy leakage in personalized mobile In-App Ads," in *Proc. 2nd Netw. Distrib. Syst. Security Symp.*, 2016.
- [4] H. Li, H. Zhu, S. Du, X. Liang, and X. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," *IEEE Trans. Dependable Secure Comput.*, vol. PP, no. 99, p. 1, 2016.
- [5] H. Li, Z. Xu, H. Zhu, D. Ma, S. Li, and K. Xing, "Demographics inference through Wi-Fi network traffic analysis," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2016, pp. 1–9.
- [6] H. Li, H. Zhu, and D. Ma, "Demographic information inference through meta-data analysis of Wi-Fi traffic," *IEEE Trans. Mobile Comput.*, vol. PP, no. 99, p. 1, 2017.

- [7] J. Liu, J. Liu, H. Li, H. Zhu, N. Ruan, and D. Ma, "Who moved my cheese: Towards automatic and fine-grained classification and modeling Ad network," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–6.
- [8] P. Wang, Z. Gao, X. Xu, Y. Zhou, H. Zhu, and K. Q. Zhu, "Automatic inference of movements from contact histories," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 386–387, 2010.
- [9] A. M. Vegni and V. Loscri, "A survey on vehicular social networks," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2397–2419, Oct.–Dec. 2015.
- [10] N. Korula and S. Lattanzi, "An efficient reconciliation algorithm for social networks," *Proc. VLDB Endowment*, vol. 7, no. 5, pp. 377–388, 2014.
- [11] Z. Zhang, Q. Gu, T. Yue, and S. Su, "Identifying the same person across two similar social networks in a unified way: Globally and locally" *Inf. Sci.*, vol. 394, pp. 53–67, 2017.
- [12] P. Pedarsani, D. R. Figueiredo, and M. Grossglauser, "A Bayesian method for matching two similar graphs without seeds," in *Proc. 51st Annu. Allerton Conf. Commun. Control Comput.*, 2013, pp. 1598–1607.
- [13] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *Proc. 30th IEEE Symp. Security Privacy*, 2009, pp. 173–187.
- [14] S. Nilizadeh, A. Kapadia, and Y. Y. Ahn, "Community-enhanced de-anonymization of online social networks," in *Proc. ACM Conf. Comput. Commun. Security*, 2014, pp. 537–548.
- [15] S. Lai, H. Li, H. Zhu, and N. Ruan, "De-anonymizing social networks: Using user interest as a side-channel," in *Proc. IEEE/CIC Int. Conf. Commun. China*, 2015, pp. 1–5.
- [16] S. Ji, W. Li, M. Srivatsa, J. He, and R. Beyah, "Structure based data de-anonymization of social networks and mobility Traces," in *Information Security*, Berlin, Germany: Springer, 2014, 237–254.
- [17] M. Srivatsa and M. Hicks, "Deanonymizing mobility traces: Using social networks as a side-channel," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 628–637.
- [18] S. Ji, W. Li, M. Srivatsa, J. He, and R. Beyah, "General graph data de-anonymization: From mobility traces to social networks," *ACM Trans. Inf. Syst. Security*, vol. 18, no. 4, 2016, Art. no. 12.
- [19] S. Ji, W. Li, N. Gong, P. Mittal, and R. Beyah, "On your social network de-anonymizability: Quantification and large scale evaluation with seed knowledge," in *Proc. 2nd Netw. Distrib. Syst. Security Symp.*, 2015.
- [20] S. Ji, W. Li, N. Gong, P. Mittal, and R. Beyah, "Seed-based de-anonymizability quantification of social networks," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 7, pp. 1398–1411, Jul. 2016.
- [21] S. Ji, W. Li, P. Mittal, X. Hu, and R. Beyah, "SecGraph: A Uniform and open-source evaluation system for graph data anonymization and de-anonymization," in *Proc. 24th USENIX Conf. Security Symp.*, 2015, pp. 303–318.
- [22] X.-Y. Li, C. zhang, T. Jung, J. Qian, and L. Chen, "Graph-based privacy-preserving data publication," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
- [23] J. Qian, X.-Y. Li, C. Zhang, and L. Chen, "De-anonymizing social networks and inferring private attributes using knowledge graphs," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
- [24] H. Fu, A. Zhang, and X. Xie, "Effective social graph deanonymization based on graph structure and descriptive information," *ACM Trans. Intell. Syst. Technol.* vol. 6, no. 4 pp. 49:1–49:29, 2015.
- [25] T. Iofciu, P. Fankhauser, F. Abel, and K. Bischoff, "Identifying users across social tagging systems," in *Proc. Int. Conf. Web Soc. Media*, 2011.
- [26] O. Peled, M. Fire, L. Rokach, and Y. Elovici, "Matching entities across online social networks," *Neurocomputing*, vol. 210, pp. 91–106, 2016.
- [27] O. Goga, H. Lei, S. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 447–458.
- [28] J. Vosecky, D. Hong, and Y. Shen, "User identification across multiple social networks," in *Proc. IEEE 1st Int. Conf. Netw. Digit. Technol.*, 2009, pp. 360–365.
- [29] R. Zafarani and H. Liu, "Connecting users across social media sites: A behavioral-modeling approach," in *Proc. 19th ACM Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 41–49.
- [30] M. Korayem and D. Crandall, "De-anonymizing users across heterogeneous social computing platforms," in *Proc. Int. Conf. Web Soc. Media*, 2013.
- [31] G. Wondracek, et al., "A practical attack to de-anonymize social network users," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 223–238.
- [32] Y. Zhang, J. Tang, Z. Yang, J. Pei, and S. Yu, "COSNET: connecting heterogeneous social networks with local and global consistency," in *Proc. 21th ACM Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1485–1494.
- [33] H. Li, Q. Chen, H. Zhu, and D. Ma, "Hybrid de-anonymization across real-world heterogeneous social networks," in *Proc. ACM Turing 50th Celebration Conf.-China*, 2017, Art. no. 33.
- [34] M. Li, N. Cao, S. Yu, and W. Lou, "FindU: privacy-preserving personal profile matching in mobile social network," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2011, pp. 2435–2443.
- [35] M. Li, S. Yu, N. Cao, and W. Lou, "Privacy-preserving distributed profile matching in proximity-based mobile social networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 5, pp. 2024–2033, May 2013.
- [36] R. Zhang, Y. Zhang, J. Sun, and G. Yan, "Fine-grained private matching for proximity-based mobile social networking," in *Proc. IEEE Conf. Inf. Comput. Commun.*, 2012, pp. 1969–1977.
- [37] R. Zhang, Y. Zhang, J. Sun, and G. Yan, "Privacy-preserving profile matching for proximity-based mobile social networking," *IEEE J. Select. Areas Commun.*, vol. 31, no. 9, pp. 656–668, Sep. 2013.
- [38] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Academy Sci.*, vol. 105, no. 4, pp. 1118–1123, 2008.
- [39] A. E. Monge and E. Charles, "The field matching problem: Algorithms and applications," *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 267–270.
- [40] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," in *Proc. Workshop Data Cleaning Object Consolidation*, 2003, vol. 3, pp. 73–78.
- [41] GeoNames. 17 Jun. 2009. [Online]. Available: <http://geonames.org/>
- [42] D. Perito, C. Castelluccia, M. A. Kaafar, and P. Manils, "How unique and traceable are usernames?" in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2011, pp. 1–17.
- [43] L. Backstrom, C. Dwork, and J. Kleinberg, "Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 181–190.
- [44] L. Yartseva and M. Grossglauser, "On the performance of percolation graph matching," in *Proc. 1st ACM Conf. Online Soc. Netw.*, 2013, pp. 119–130.
- [45] X. Kong, J. Zhang, and S. Y. Philip, "Inferring anchor links across multiple heterogeneous social networks," in *Proc. Conf. Inf. Knowl. Manage.*, 2013, pp. 179–188.
- [46] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani, "Sigma: Simple greedy matching for aligning large knowledge bases," in *Proc. 19th ACM Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 572–580.
- [47] J. Zhang, J. Sun, R. Zhang, and Y. Zhang, "Your actions tell where you are: Uncovering twitter users in a metropolitan area," in *Proc. IEEE Conf. Commun. Netw. Security*, 2015, pp. 424–432.



Huaxin Li received the BSc degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2011. He is working towards the MSc degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include social networks privacy, smartphone security, network security and privacy, and machine learning.



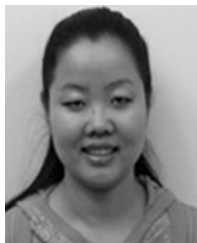
Qingrong Chen is currently working toward the bachelor's degree with Shanghai Jiao Tong University. His research interests include social network security, Bitcoin security, and network privacy.



Haojin Zhu (IEEE M'09-SM'16) received the BSc degree from Wuhan University (China), the MSc degree from Shanghai Jiao Tong University (China), both in computer science and the PhD degree in electrical and computer engineering from the University of Waterloo (Canada), in 2002, 2005, and 2009. Since 2017, he has been a full professor with Computer Science Department, Shanghai Jiao Tong University. His current research interests include network security and privacy enhancing technologies. He published 35 international journal papers, including the *IEEE Journal on Selected Areas in Communications*, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Mobile Computing*, the *IEEE Transactions on Wireless Communications*, the *IEEE Transactions on Vehicular Technology*, and 60 international conference papers, including ACM CCS, ACM MOBICOM, ACM MOBIHOC, IEEE INFOCOM, IEEE ICDCS. He received a number of awards including: IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award (2014), Top 100 Most Cited Chinese Papers Published in International Journals (2014), Supervisor of Shanghai Excellent Master Thesis Award (2014), Distinguished member of the IEEE INFOCOM Technical Program Committee (2015), Outstanding Youth Post Expert Award for Shanghai Jiao Tong University (2014), SMC Young Research Award of Shanghai Jiao Tong University (2011). He was a co-recipient of best paper awards of the IEEE ICC (2007) and Chinacom (2008) as well as the IEEE GLOBECOM Best Paper Nomination (2014). He received Young Scholar Award of Changjiang Scholar Program by Ministry of Education of P.R. China in 2016. He is a senior member of the IEEE.



Di Ma received the PhD degree from the University of California, Irvine, in 2009. She is an associate professor in the Computer and Information Science Department, the University of Michigan-Dearborn, where she leads the Security and Forensics Research Lab (SAFE). She is broadly interested include the general area of security, privacy, and applied cryptography. Her research spans a wide range of topics, including smartphone and mobile device security, RFID and sensor security, vehicular network and vehicle security, computation over authenticated/encrypted data, fine-grained access control, secure storage systems, and so on. Her research is supported by NSF, NHTSA, AFOSR, Intel, Ford, and Research in Motion. She was with IBM Almaden Research Center in 2008 and the Institute for Infocomm Research, Singapore in 2000-2005. She won the Tan Kah Kee Young Inventor Award in 2004. She is a member of the IEEE.



Hong Wen received the MSc degree from the Sichuan Union University of Sichuan, P. R. China, in 1997, and the PhD degree from the Communication and Computer Engineering Department, the Southwest Jiaotong University (Chengdu, P. R. China). Then she worked as an associate professor in the National Key Laboratory of Science and Technology on Communications at UESTC, P. R. China. From January 2008 to August 2009, she was a visiting scholar and postdoctoral fellow in the ECE Department, University of Waterloo.

Now she holds the professor position at UESTC, P. R. China. Her major interests focus on wireless communication system security. She is a member of the IEEE.



Xuemin (Sherman) Shen (IEEE M'97-SM'02-F09) received the BSc degree from Dalian Maritime University China, the MSc and PhD degrees from Rutgers University, New Jersey, all in electrical engineering, in 1982, 1987, and 1990. He is a professor and University research chair, in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is also the associate chair for Graduate Studies. His research focuses on resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is an elected member of the IEEE ComSoc Board of Governor, and the Chair of Distinguished Lecturers Selection Committee. He served as the Technical Program Committee chair/co-chair of the *IEEE Globecom'16*, the *IEEE Infocom'14*, the *IEEE VTC'10 Fall*, and the *IEEE Globecom'07*, the Symposia chair of the IEEE ICC'10, the Tutorial Chair for IEEE VTC'11 Spring and IEEE ICC'08, the General co-chair of the *ACM Mobihoc'15*, *Chinacom'07* and *QShine'06*, the chair of the *IEEE Communications Society Technical Committee on Wireless Communications*, and the *P2P Communications and Networking*. He also serves/served as the editor-in-chief of the *IEEE Network*, *Peer-to-Peer Networking and Application*, and *IET Communications*; a Founding Area editor of the *IEEE Transactions on Wireless Communications*; an associate editor of the *IEEE Transactions on Vehicular Technology*, *Computer Networks*, and the *ACM/Wireless Networks*, etc.; and the guest editor of the *IEEE Journal on Selected Areas in Communications*, the *IEEE Wireless Communications*, the *IEEE Communications Magazine*, and the *ACM Mobile Networks and Applications*, etc. He received the Excellent Graduate Supervision Award in 2006, and the Outstanding Performance Award in 2004, 2007, 2010, and 2014 from the University of Waterloo, the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada, and the Distinguished Performance Award in 2002 and 2007 from the Faculty of Engineering, University of Waterloo. He is a registered Professional Engineer of Ontario, Canada, an the fellow of the IEEE, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer of IEEE Vehicular Technology Society and Communications Society.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Privacy-Preserving Data Processing with Flexible Access Control

Wenxiu DING¹, Zheng Yan¹, *Senior Member, IEEE*, and Robert H. Deng², *Fellow, IEEE*

Abstract—Cloud computing provides an efficient and convenient platform for cloud users to store, process and control their data. Cloud overcomes the bottlenecks of resource-constrained user devices and greatly releases their storage and computing burdens. However, due to the lack of full trust in cloud service providers, the cloud users generally prefer to outsource their sensitive data in an encrypted form, which, however, seriously complicates data processing, analysis, as well as access control. Homomorphic encryption (HE) as a single key system cannot flexibly control data sharing and access after encrypted data processing. How to realize various computations over encrypted data in an efficient way and at the same time flexibly control the access to data processing results has been an important challenging issue. In this paper, we propose a privacy-preserving data processing scheme with flexible access control. With the cooperation of a data service provider (DSP) and a computation party (CP), our scheme, based on Paillier's partial homomorphic encryption (PHE), realizes seven basic operations, i.e., *Addition, Subtraction, Multiplication, Sign Acquisition, Absolute, Comparison, and Equality Test*, over outsourced encrypted data. In addition, our scheme, based on the homomorphism of attribute-based encryption (ABE), is also designed to support flexible access control over processing results of encrypted data. We further prove the security of our scheme and demonstrate its efficiency and advantages through simulations and comparisons with existing work.

Index Terms—Homomorphic encryption, privacy preservation, data sharing, attribute-based encryption

1 INTRODUCTION

CLOUD computing has been widely adopted in various application domains owing to its specific advantages. It enables cloud users to store their data and perform various computations on the data without incurring a high cost. With the advent of Internet of Things, enormous amounts of data are produced and outsourced to the cloud or cloudlets for storage and analysis. Data analysis helps to gain insights on related entities in a physical world, which can provide tremendous values to various applications in multifarious domains (e.g., medical [1] and business [2]).

However, the cloud may not be fully trusted by cloud users since it may reveal or disclose the data outsourced by the cloud users or their processing results, which may seriously impact user privacy. For example, medical case analysis can help in predicting potential illness, but patients may be reluctant to provide their health data due to privacy concerns. Therefore, it is of great significance to protect sensitive data and data processing results from being leaked to any unauthorized parties. A standard solution is to encrypt the

data before uploading them to the cloud. However, data encryption introduces several challenges as described below.

First, encryption seriously restricts computations and analyses over the outsourced data in the cloud. With traditional encryption algorithms (e.g., AES), it is impossible for the cloud to process the encrypted data directly. Some existing efforts adopted partial homomorphic encryption (PHE) to solve the problem, but they are limited to multiplication and addition operations on encrypted data [3], [4], which cannot satisfy the demands of many applications. More operations, such as sign acquisition, comparison, absolute and equality test, are expected to be supported in practice [5], [6]. This requests further study on privacy-preserving computations. More basic operations over ciphertexts can obviously support more applications that apply different functions and algorithms, e.g., privacy-preserving classifications in machine learning [7], trust evaluation in Internet of Things [8], and medical analysis in e-health [9]. To realize arbitrary computations over ciphertexts, schemes based on fully homomorphic encryption (FHE) were designed [10], [11], [12]. However, most FHE based schemes suffer from huge computational overhead and high storage cost, which make them impractical for real world deployment and wide use. Currently, the literature still lacks serious studies on efficient computations over ciphertexts.

Second, multi-user access control over ciphertext processing results should also be supported [13]. Existing PHE and FHE schemes are both single-user systems, which inherently lack support on multi-user access to the processing results of encrypted data. The scheme based on PHE [14] supports distribution of addition operation results through an interactive protocol between two servers. But

- W. Ding is with the State Key Laboratory on Integrated Services Networks, School of Cyber Engineering, Xidian University, Chang'an Qu 710126, China. E-mail: wenxiuding_1989@126.com.
- Z. Yan is with the State Key Lab on Integrated Services Networks, School of Cyber Engineering, Xidian University, No.2 South Taibai Road, Xi'an 710071 China, and with the Department of Communications and Networking, Aalto University, Konemiehentie 2, P.O.Box 15400, Espoo 02150, Finland. E-mail: zyan@xidian.edu.cn.
- R.H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: robertdeng@smu.edu.sg.

Manuscript received 22 July 2017; revised 27 Nov. 2017; accepted 7 Dec. 2017. Date of publication 22 Dec. 2017; date of current version 18 Mar. 2020. (Corresponding author: Zheng Yan.)
Digital Object Identifier no. 10.1109/TDSC.2017.2786247

this protocol must be executed for each data request, thus it is inefficient. Attribute-based encryption (ABE) is an effective tool to support fine-grained access control and multi-user access. It has been applied in many application scenarios [15], [16], [17]. However, to our knowledge, there is no effort in the literature on fine-grained access control over the computation/analysis results based on encrypted data. Our previous work [18] aims to solve this problem by combining homomorphic encryption and proxy re-encryption, but it only supports one requester access at one time. In case multiple users want to access the same result, it needs to execute the designed scheme for each requester one by one, which obviously incurs high communication and computation costs, as shown in experiments in Section 5.2.3.

In this paper, we propose a novel scheme in order to overcome the challenges as described above. It supports multiple basic computations over encrypted data and realizes flexible access control over the processing results by employing PHE and ABE. Specifically, the contributions of this paper can be summarized as follows:

- We propose a generic system architecture consisting of a data service provider (DSP) and a computation party (CP) that seamlessly work together to simultaneously support secure computations over encrypted data and fine-grained access control of computation results.
- We present a family of protocols to efficiently realize seven basic computations over encrypted data: *Addition*, *Subtraction*, *Multiplication*, *Sign Acquisition*, *Absolute*, *Comparison*, and *Equality Test*.
- We propose to utilize ABE with homomorphism to realize fine-grained access control of the processing result of encrypted data, which is not revealed to any system entities including DSP and CP.
- We prove the security of the proposed scheme and demonstrate its efficiency through simulations and comparisons with existing schemes. We show that the proposed scheme is suitable for big data processing. It can be applied in any scenarios with either a small or a large number of data providers.

The rest of this paper is organized as follows. Section 2 gives a brief overview of related work. Section 3 introduces the system model and attack model of our proposed scheme, followed by its detailed design in Section 4. In Section 5, security analysis and performance evaluation are given. Finally, a conclusion is presented in the last section.

2 RELATED WORK

With the development of cloud computing, cloud users benefit from outsourcing data storage and computation to the cloud. However, the risk of personal data disclosure makes it urgent to enhance data security and user privacy. Besides those schemes focusing on data aggregation [19], [20], [21], [22], [23], [24], [25], other studies were conducted to achieve more efficient privacy-preserving operations. In addition, many constructions have been proposed to realize secure access control although the aforementioned issues are still open.

2.1 Secure Data Processing Based on SMC

Secure multi-party computation (SMC) enables computations over multi-user outsourced data without revealing

any input. It lays a technical foundation for many problems, such as database query, intrusion detection and data mining with privacy preservation [8]. Several schemes [26], [27] based on the popular SMC construction Sharemind [28] were proposed to achieve various secure computations, e.g., multiplication. But the product of N pieces of data needs about 3^N multiplications of 32-bit numbers under the cooperation of three involved servers, which obviously cannot adapt to big data processing. Although a data requester can easily obtain the final result by requesting all secret pre-processing shares from all involved servers, how to realize fine-grained access control in SMC is still an open issue.

2.2 Secure Data Processing Based on Homomorphic Encryption

FHE schemes [10], [11], [12] are designed to realize arbitrary computations over encrypted data. Due to high computation overhead, some extended schemes [29], [30] were proposed to improve FHE efficiency. However, the computation and storage costs of existing schemes are still not satisfactory for practical use [31], [32].

PHE has been widely used in many applications because it is more efficient and practical than FHE although it can only support limited computations. Some schemes [3], [4] were proposed to support more types of computations, but they can only support addition and multiplication over a limited number of data inputs. In [3], decryption requires solving the problem of discrete logarithm, which seriously restricts the length and the number of data inputs. The multiparty computation framework proposed in [4] achieves addition and multiplication by applying secret sharing. Similar to the SMC-based scheme in [27], it is unable to support the multiplication of a large number of data inputs. Liu et al. [5] proposed a framework for efficient outsourced data calculations with privacy preservation, which can deal with several types of operations, such as addition, multiplication, and division. But this framework cannot support multiplication of a large number of data.

Besides the problems mentioned above, the biggest problem of PHE is that it is a single-user system. This means that the data processing result based on PHE can only be decrypted and accessed by the user with the corresponding secret key of PHE. PHE is not flexible to directly support multi-user access.

2.3 Secure Data Access Control

Cloud storage enables cloud users to upload their data to the cloud for storage and further sharing. However, this causes a new problem that the cloud users lose full control over their data. Thus, an efficient and secure data access control scheme is needed. A number of solutions have been proposed to protect the outsourced data in the cloud, as briefly introduced below.

Proxy re-encryption was adopted to manage data sharing in cloud [33], [34]. But it cannot support fine-grained access control on homomorphic computation results. Role-based access control (RBAC) can only provide partial flexibility based on one level policy, which ensures that only the user with a specified role can access data. But, the constructions [35], [36] based on RBAC cannot support flexible access policies described with various attribute structures. ABE [37],

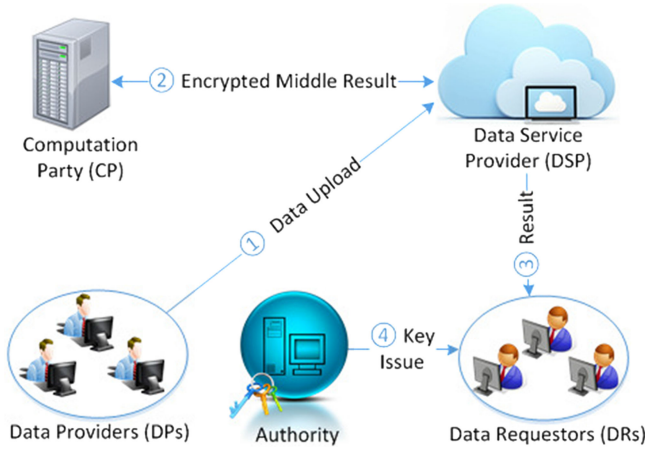


Fig. 1. A system model.

[38] was widely applied in cloud storage management for achieving fine-grained access control [39], [40], [41]. Furthermore, trust-based schemes [15], [16], [17] simplify the attributes involved in ABE and take into consideration only trust levels. These schemes highly reduce computational costs. But, only one system entity (such as one user or one server) is in charge of the access control, which makes this entity obviously knows the contents of results. Thus, it cannot satisfy the demand that the final result cannot be accessed by any unauthorized entity including servers. In this paper, we propose a scheme by taking advantage of the homomorphism of ABE under the same policy to control the access of the processing result of encrypted data based on the cooperation of two servers.

3 PROBLEM STATEMENTS

3.1 System Model

Our proposed system mainly comprises five types of entities as shown in Fig. 1:

- 1) Data service provider (DSP) stores user data, provides some computation service and controls user access. DSP can be operated by a public cloud provider.
- 2) Computation party (CP) bears the responsibility of computations and access control. It can be served by a private cloud service provider or an administrative department of a company or an institute, which fulfills partial computations and controls access. There may exist multiple CPs for different applications. Each CP provides services for its own consumers. Herein, we simplify our design by considering only one CP in this paper.
- 3) Data providers (DPs) are the data collectors or producers that encrypt data and store ciphertexts in the DSP for storage and processing.
- 4) Data requestors (DRs) are the data consumers that acquire the result of data processing/analyzing in a specific context. A DR can also be a DP. DRs are cloud users that take advantage of cloud computing in terms of data storage and data computation. Since the provided data are encrypted, the data processing result is also in an encrypted form. This raises the issue of data access control with regard to the processing results of encrypted data.

- 5) Authority is fully trusted, which is responsible for system parameter generation and ABE key issuing.

The DPs provide their personal data in an encrypted form and store them at the DSP. Then the DSP cooperates with the CP to complete basic computations over the collected data. In addition, the DSP and CP together execute the access control over the final result of data processing. Only those DRs that satisfy a specific policy can access the final result with the key issued by Authority.

3.2 Attack Model

In the above system, the Authority is regarded as fully trusted to perform its duties, which acts honestly and would never collude with any other entities. All other entities are curious-but-honest. That is, they are curious about others' data, but act honestly by strictly following the design of the system. The DSP and the CP would never collude with each other due to conflict of business interests (e.g., consumer resources and market division) and their legal responsibility. Any their collusion will make them lose reputation, which finally impacts their profits. Herein, we introduce an adversary \mathcal{A}^* in our model, which aims to obtain some specific data by challenging a cloud user (either a DR or a DP) with following capabilities:

- 1) \mathcal{A}^* may eavesdrop all communication channels to access any encrypted data except the channels between Authority and DRs;
- 2) \mathcal{A}^* may compromise the DSP (or the CP) to guess the raw data of all ciphertexts outsourced from the DPs, and the raw data of all ciphertexts sent to the CP (or the DSP) and the DR;
- 3) \mathcal{A}^* may compromise the DSP (or the CP) together with the DPs to guess the final data processing result.
- 4) \mathcal{A}^* may compromise the DSP (or the CP) together with the DR to guess the raw data from the DP.

The attack model has one restriction: \mathcal{A}^* cannot compromise the challenged DR or DP. The adversary would like to know the raw data of DP (by attacking the DP) or the processing result (by attacking the DR).

4 PRIVACY-PRESERVING DATA PROCESSING WITH ACCESS CONTROL

4.1 Notations and Preliminaries

4.1.1 Notations

For easy presentation, Table 1 summarizes the notations used in this paper.

4.1.2 Additive Homomorphic Encryption

Paillier's cryptosystem [42] is one of the most important additive homomorphic encryption. Suppose we have N pieces of encrypted data under same key pk , which can be presented as $[m_i]_{pk}$ ($i = 1, 2, \dots, N$). The additive homomorphic encryption satisfies the following equation:

$$D_{sk}\left(\prod_{i=1}^N [m_i]_{pk}\right) = \sum_{i=1}^N m_i,$$

where $D_{sk}()$ is the corresponding homomorphic decryption algorithm with secret key sk .

TABLE 1
Notation Description

Symbols	Description
g	The system generator that is public;
n	The system parameter;
(sk_{DSP}, pk_{DSP})	The key pair of DSP for data processing;
(sk_{CP}, pk_{CP})	The key pair of CP for data processing;
$PK = pk_{DSP}^{sk_{CP}} = pk_{CP}^{sk_{DSP}}$	The public parameter based on keys of DSP and CP;
m_i	The raw data provided by DP i ;
$[m]$	The ciphertext of m under PK ;
$[m]_{pk_i}$	The ciphertext of m under public key pk_i ;
\hat{m}	The masked message of m ;
f	The sign flag in <i>Sign Acquisition, Comparison, and Equality Test</i> ;
r	The random value;
N	The number of data providers;
$\mathcal{L}(\ast)$	The bit length of input data;
(ck, pk_{ck})	The key pair for final access control;
CK'	The ciphertext of ck using ABE;
ck_1	The partial key for access control chosen by DSP;
ck_2	The partial key for access control chosen by CP;
MSK'	The master secret key in ABE;
SK'	The decryption key in ABE;
PK'	The public key in ABE

4.1.3 Key-Policy Attribute-Based Encryption (KP-ABE)

In KP-ABE, ciphertexts are generated based on some descriptive attributes while decryption keys are associated with policies. Generally, KP-ABE [38] consists of four algorithms: *Setup*, *Encrypt*, *KeyGen*, and *Decrypt*.

Setup^{ABE}(λ, U) \rightarrow (PK' , MSK'). The setup algorithm takes in security parameter λ and attribute universe description $U = \{1, 2, \dots, \omega\}$. It outputs public parameters PK' ($T_1 = g^1, \dots, T_{|U|} = g^{|U|}, Y = e(g, g)^y$) and master secret key $MSK'(t_1, \dots, t_{|U|}, y)$, where g' is the generator of G_1 with $e: G_1 \times G_1 \rightarrow G_T$.

Enc^{ABE}(M, γ, PK') \rightarrow CK' . The encryption algorithm takes in message M , a set of attributes γ and PK' . It outputs ciphertext $CK'(\gamma, E' = MY^s, \{E_i = T_i^s\}_{i \in \gamma})$, where s is a randomly chosen number.

KeyGen^{ABE}(\mathcal{T}, MSK') \rightarrow SK' . The key generation algorithm takes in access structure \mathcal{T} , the master secret key MSK' . It outputs decryption key SK' ($SK'_i = g^{q_x(0)/t_i}$ where $i = att(x)$, $q_x(0) = q_{parent(x)}(index(x))$) and $q_r(0) = y$.

Dec^{ABE}(CK', PK', SK') \rightarrow M . The decryption algorithm takes in PK' , SK' and the ciphertext CK' . If the set of attributes satisfies the access policy tree \mathcal{T} embedded in the private key, it finally outputs the message M .

For more details about KP-ABE, refer to [38]. Notably, ciphertext-policy attribute-based encryption (CP-ABE) [37] can also be applied to implement our scheme. The KP-ABE is multiplicative homomorphic if the same attributes are employed to encrypt two pieces of raw data. That is, given two ABE ciphertexts of M_1 and M_2 under the same policy, the ciphertext of $M_1 * M_2$ can be obtained through the multiplication of two ciphertexts $Enc^{ABE}(M_1, \gamma, PK') * Enc^{ABE}(M_2, \gamma, PK')$, marked as HE^{ABE} . The length of raw data is limited and highly related to the system parameters. In this paper, we employ this feature to realize the secure access control that can prevent the reveal of processing

result to any involved entities. Refer to Section 5.2.3 for more details.

4.2 Homomorphic Re-Encryption Scheme (HRES)

In order to support privacy-preserving data processing, we revise the scheme [43] (named as EDD) and design the HRES to provide two-level decryption and achieve secure data processing. The complete version of HRES is introduced in our previous work [18].

Key Generation (KeyGen). Let k be a security parameter and p, q be two large primes, where $\mathcal{L}(p) = \mathcal{L}(q) = k$ ($\mathcal{L}(\cdot)$ returns the bit length of input data). Due to the property of safe primes, there exist two primes p' and q' which satisfy that $p = 2p' + 1$, $q = 2q' + 1$. We compute $n = p * q$ and choose a generator g with order $\lambda = 2p'q'$, which can be chosen by selecting a random number $z \in \mathbb{Z}_{n^2}^*$ and computing $g = -z^{2n}$. The value λ can be used to decrypt the encrypted data, but we choose to conceal it and protect it from all involved parties. In the HRES, we only use key pair (sk, g^{sk}) for data encryption and decryption. The DSP and the CP generate their key pairs: $(sk_{DSP} = a, pk_{DSP} = g^a)$ and $(sk_{CP} = b, pk_{CP} = g^b)$, and then negotiate their Diffie-Hellman key $PK = pk_{DSP}^{sk_{CP}} = pk_{CP}^{sk_{DSP}} = g^{a*b}$. To support encrypted data processing, PK is public to all involved parties. At system setup, cloud user i (i.e., DP or DR) generates its key pair $(sk_i, pk_i) = (k_i, g^{k_i})$. The public system parameters include $\{g, n, PK\}$.

First, the original encryption is directly obtained from [43], which is a general public key cryptosystem.

Encryption (Enc). Any entity (e.g., cloud user or cloud server) wants to send its data to a specific cloud user i . It simply encrypts its data with the public key of cloud user i (pk_i) and random $r \in [1, n/4]$, and sends ciphertext to cloud user i :

$$[m]_{pk_i} = \{(1 + m * n)pk_i^r, g^r\} \pmod{n^2}.$$

Decryption (Dec). Upon receiving the encrypted data, cloud user i can directly decrypt it to obtain the original data:

$$m = L\left((1 + m * n)pk_i^r / (g^r)^{k_i} \pmod{n^2}\right).$$

Second, a *Two-Level Decryption* scheme is newly designed to flexibly support outsourced data processing, as presented below:

Encryption with Two Keys (EncTK). Given message $m_i \in \mathbb{Z}_n$ provided by cloud user i , we first select random number $r \in [1, n/4]$ and then encrypt it with PK generated from the keys of two servers. The ciphertext is generated as $[m_i] = [m_i]_{PK} = \{T_i, T_i^r\}$, where $T_i = (1 + m_i * n) * PK^r \pmod{n^2}$ and $T_i^r = g^r \pmod{n^2}$.

Note: for ease of presentation, we use $[m_i]$ to denote the ciphertext of m_i encrypted with PK , which can only be decrypted under the cooperation of DSP and CP.

Partial Decryption with SK_{DSP} (PDec1). Once $[m_i]$ is received by the DSP, algorithm *PDec1* will be run to transfer it into another ciphertext that can be decrypted by the CP as follows:

$$\begin{aligned} [m_i]_{pk_{CP}} &= \{T_i^{(1)}, T_i^{r(1)}\} = \{T_i, (T_i^r)^{sk_{DSP}}\} \\ &= \{(1 + m_i * n)PK^r, g^{r*a}\} \pmod{n^2} \\ &= \{(1 + m_i * n)pk_{CP}^{a*r}, g^{r*a}\} \pmod{n^2}. \end{aligned}$$

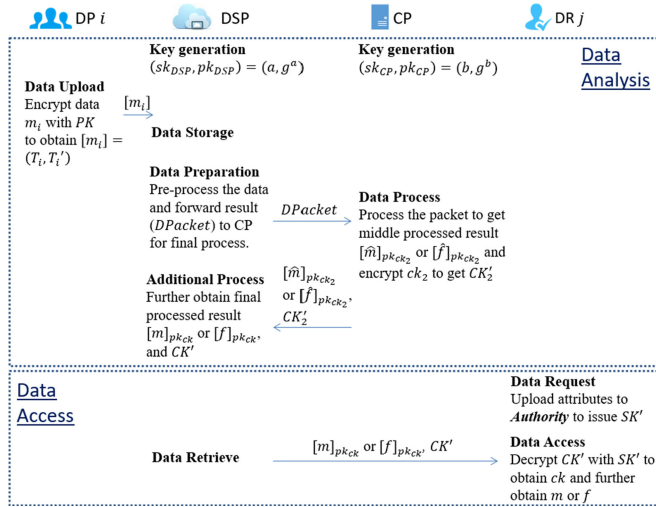


Fig. 2. A brief procedure of data processing.

Partial Decryption with SK_{CP} (PDec2). Once the encrypted data $[m_i]_{pk_{CP}}$ is received, the CP can directly decrypt it with its own secret key as follows:

- 1) $T_i'^{(2)} = (T_i'^{(1)})^{sk_{CP}} = g^{r*a*b} = PK^r \bmod n^2$;
- 2) $m_i = L(T_i'^{(1)}/T_i'^{(2)} \bmod n^2)$, where $L(u) = (u - 1)/n$.

In addition, the encrypted data under any public key pk has the following properties:

- 1) Additive homomorphism:
 $[m_1]_{pk} * [m_2]_{pk} = [m_1 + m_2]_{pk}$;
- 2) $([m]_{pk})^t = \{ \{ (1 + m*n)pk^r \}^t, (g^r)^t \} \bmod n^2$
 $= \{ (1 + m*n)^t pk^{r*st}, g^{r*st} \} \bmod n^2$
 $= \{ (1 + t*m*n)pk^{r*st}, g^{r*st} \} \bmod n^2$
 $= [t*m]_{pk}$;
- 3) $([m]_{pk})^{n-1} = \{ \{ (1 + m*n)pk^r \}^{(n-1)}, (g^r)^{(n-1)} \} \bmod n^2$
 $= \{ (1 + m*(n-1)*n)pk^{r(n-1)}, g^{r(n-1)} \} \bmod n^2$
 $= \{ (1 - m*n + m*n^2)pk^{r(n-1)}, g^{r(n-1)} \} \bmod n^2$
 $= \{ (1 - m*n)pk^{r(n-1)}, g^{r(n-1)} \} \bmod n^2$
 $= [-m]_{pk}$.

4.3 Data Processing Procedure

The brief procedure of data processing is shown in Fig. 2. It mainly involves six steps:

Step 1 (System Setup @ All Entities). Authority calls the algorithm *KeyGen* in Section 4.2 and *Setup^{ABE}(λ, U)* in Section 4.1.3 to complete the setup of HRES and ABE. Note: if multiple CPs are employed in the system, each CP can negotiate a Diffie-Hellman key PK with the DSP and publish this key to its customers.

Step 2 (Data Upload @ DPs). DPs encrypt their personal data before uploading it to the DSP. It directly recalls *EncTK* to encrypt data m_i (Unless otherwise specified, $|m_i| < \mathcal{L}(n)/4$):

$$[m_i] = (T_i, T_i') = \{ (1 + m_i * n) * PK^{T_i}, g^{T_i} \} \bmod n^2.$$

Step 3 (Data Preparation @ DSP). Upon receiving the data from DPs, the DSP needs to do some analyses over the encrypted data. It pre-processes the data and provides a data packet *DPacket* and ABE ciphertext for access control to the CP. The details of this process differ in various

operations and are given in the next sub-section. In addition, CP chooses a random partial key ck_1 for access control, which will be used in Step 5.

Step 4 (Data Process @ CP). Upon receiving the pre-processing results from the DSP, the CP chooses another random partial key ck_2 and further processes data to obtain the pre-processing result $[\hat{m}]_{pk_{ck_2}}$ or $[\hat{f}]_{pk_{ck_2}}$.

Regarding to access control, CP encrypts ck_2 using ABE to get $CK'_2 = Enc^{ABE}(ck_2, \gamma, PK')$ and forwards it to DSP.

Step 5 (Additional Process @ DSP). The DSP needs to further remove the mask from ciphertext $[\hat{m}]_{pk_{ck_2}}$ or $[\hat{f}]_{pk_{ck_2}}$ to obtain the final processing ciphertext $[m]_{pk_{ck}}$ or $[f]_{pk_{ck}}$ where $pk_{ck} = g^{ck} = g^{ck_1 * ck_2}$ and $ck = ck_1 * ck_2$.

Regarding to access control, the DSP encrypts ck_1 using ABE under the same policy to get CK'_1 and further gets CK' through the homomorphism of ABE:

$$CK' = CK'_1 * CK'_2 = Enc^{ABE}(ck_1 * ck_2, \gamma, PK').$$

Finally, the DSP keeps $[m]_{pk_{ck}}$ or $[f]_{pk_{ck}}$, and CK' for user access.

Step 6 (Data Access @ DR). If the DR satisfies the access policy, Authority issues a secret key SK' to the DR. Hence, the DR can decrypt CK' to get ck , and further obtain m or f to meet their computation demand.

4.4 Detailed Data Processing

System setup and data collection are the same as those in Section 4.3. The operations for access control are also the same as those above, thus we do not introduce the details in this section.

Our proposed scheme supports seven basic operations over encrypted data: 1) *Addition*; 2) *Subtraction*; 3) *Multiplication*; 4) *Sign Acquisition*; 5) *Absolute*; 6) *Comparison*; and 7) *Equality Test*. In the following sub-sections, we mainly focus on the steps from 3 to 5 in each basic operation.

4.4.1 Addition

Addition obtains the sum of all raw data: $m = \sum_{i=1}^N m_i$, which can be accomplished by multiplying all ciphertexts. Note that the number of the data in *Addition* affects the length of the provided data. If we want to get the sum result of N pieces of data, it should guarantee that $m_i < n/N$.

Step 3 (Data Preparation @ DSP). Due to additive homomorphism, the DSP can directly multiply encrypted data one by one as follows:

$$[m] = (T, T') = \prod_{i=1}^N [m_i] = \left(\prod_{i=1}^N T_i, \prod_{i=1}^N T_i' \right)$$

To realize group access control, it chooses a random number r_1 and the first partial key ck_1 , and then computes as follows:

- 1) Compute $c_1 = ck_1^{-1} \bmod n^2$;
- 2) Mask ciphertext:

$$[c_1(m + r_1)] = \left(\tilde{T}, \tilde{T}' \right) = \{ (T(1 + r_1 * n))^{c_1}, (T')^{c_1} \}$$

- 3) Call *PDec1* to partially decrypt it:

$$[c_1(m + r_1)]_{pk_{CP}} = \left(\hat{T}, \hat{T}' \right) = \left\{ \hat{T}, \left(\hat{T}' \right)^a \right\}$$

Then DSP sends $[c_1(m+r_1)]_{pk_{CP}}$ to the CP.

Step 4 (Data Process @ CP). The CP calls the algorithm *PDec2* with sk_{CP} to finally decrypt the encrypted data and obtain $c_1(m+r_1)$. And then the CP chooses the second partial key ck_2 and a random number r to encrypt data as follows:

$$[c_1(m+r_1)]_{pk_{ck_2}} = (\bar{T}, \bar{T}') = \{(1+c_1(m+r_1)n)g^{ck_2*r}, g^r\}$$

where $pk_{ck_2} = g^{ck_2}$.

Then the CP encrypts ck_2 to obtain CK'_2 and forwards $[\hat{m}]_{pk_{ck_2}}$ and CK'_2 back to the DSP.

Step 5 (Additional Process @ DSP). The DSP computes to obtain the final encrypted processing result with ck_1 and r_1 :

$$[m]_{pk_{ck_1}} = (\bar{T}^{ck_1}, \bar{T}') = \{(1+m*n)g^{ck_1*ck_2*r}, g^r\}$$

where $pk_{ck_1} = g^{ck_1*ck_2}$ and $ck_1 = ck_1*ck_2$.

Similar to Section 4.3, it encrypts ck_1 using ABE and gets $CK' = CK'_1 * CK'_2 = Enc^{ABE}(ck_1 * ck_2, \gamma, PK')$.

4.4.2 Subtraction

Subtraction obtains the subtraction of some data ($m = \sum_{i=1}^W m_i - \sum_{i=W+1}^N m_i$) with encrypted data $[m_i](i = 1, \dots, N)$. It can be accomplished by negating the subtracted terms (by raising to the power of $(n-1)$), then following the procedure of *Addition*.

Step 3 (Data Preparation @ DSP). The DSP first computes $[\sum_{i=1}^W m_i] = \prod_{i=1}^W [m_i]$ and $[\sum_{i=W+1}^N m_i] = \prod_{i=W+1}^N [m_i]$. It further calculates $[-\sum_{i=W+1}^N m_i] = ([\sum_{i=W+1}^N m_i])^{n-1}$ and multiplies them to obtain: $[m] = [(\sum_{i=1}^W m_i - \sum_{i=W+1}^N m_i)] = [\sum_{i=1}^W m_i] * [-\sum_{i=W+1}^N m_i]$. Then the subsequent process is the same to that in *Addition*. Due to length and simplicity reasons, we skip its details.

4.4.3 Multiplication

Multiplication obtains the product of all raw data ($m = \prod_{i=1}^N m_i$). Multiplication is a bit more complicated than *Addition*. It can be accomplished with three steps: 1) mask the raw data by raising to the power of a random number c ; 2) decrypt all masked ciphertexts, multiply them in the form of plaintexts and then re-encrypt masked result; 3) encrypt masked multiplication result and then remove the mask. For ease of presentation, we describe the details with two pieces of data ($[m_1], [m_2]$). The DR wants to get the multiplication result $m = m_1 * m_2$.

Note that the available number of the data in multiplication influences the length of original raw data. If we need to get the product of N pieces of data, it must be guaranteed that $\mathcal{L}(m_i) < \mathcal{L}(n)/(2N)$, which is different from *Addition*.

Step 3 (Data Preparation @ DSP). First, the DSP chooses a random partial key ck_1 and a random number c_1 , and sets another one as $c_2 = (ck_1 * c_1)^{-1} \bmod n$.

To conceal each raw data from the CP, the DSP does one exponentiation and one decryption with its own secret key by calling *PDec1*:

- 1) $[c_1 * m_1] = \{T_1^{c_1}, (T_1')^{c_1}\}$;
- 2) $[c_1 * m_1]_{pk_{CP}} = (T_1^{(1)}, T_1'^{(1)}) = \{T_1^{c_1}, (T_1')^{c_1*a}\} = \{(1+c_1*m_1*n) * PK^{T_1^{c_1}}, g^{r_1*a*c_1}\}$;
- 3) $[c_2 * m_2] = \{T_2^{c_2}, (T_2')^{c_2}\}$;

- 4) $[c_2 * m_2]_{pk_{CP}} = (T_2^{(1)}, T_2'^{(1)}) = \{T_2^{c_2}, (T_2')^{c_2*a}\} = \{(1+c_2*m_2*n) * PK^{T_2^{c_2}}, g^{r_2*a*c_2}\}$.

The data packet sent to the CP is $\{[c_1*m_1]_{pk_{CP}}, [c_2*m_2]_{pk_{CP}}\}$.

Step 4 (Data Process @ CP). Upon receiving the data packet from the DSP, the CP uses the algorithm *PDec2* to decrypt the data:

$$c_1 * m_1 = T_1^{(1)} / (T_1'^{(1)})^b, c_2 * m_2 = T_2^{(1)} / (T_2'^{(1)})^b.$$

It then chooses ck_2 and a random number r , and encrypts $c_1 * m_1 * c_2 * m_2$ and ck_2 as follows:

- 1) $[\hat{m}]_{pk_{ck_2}} = [c_1 c_2 m]_{pk_{ck_2}} = (\bar{T}, \bar{T}') = \{(1+c_1 m_1 c_2 m_2 * n) g^{ck_2*r}, g^r\}$;
- 2) $CK'_2 = Enc^{ABE}(ck_2, \gamma, PK')$.

Finally, the CP forwards $[\hat{m}]_{pk_{ck_2}}$ and CK'_2 to the DSP.

Step 5 (Additional Process @ DSP). The DSP further processes the data packet with ck_1 and then gets ciphertext of key as follows:

- 1) $[m]_{pk_{ck_1}} = \{\bar{T}^{ck_1}, \bar{T}'\} = \{(1+m*n)g^{ck_1*ck_2*r}, g^r\}$;
- 2) $CK' = CK'_2 * Enc^{ABE}(ck_1, \gamma, PK')$.

4.4.4 Sign Acquisition

We assume that $\mathcal{L}(m) < \mathcal{L}(n)/4$ and *BIG* is the largest raw data of m . Then the raw data is in the scope $[-BIG, BIG]$. DR wants to know the sign of raw data m_1 from $[m_1]$. *Sign Acquisition* can be achieved by masking the original ciphertexts with random numbers of limited length and then checking the length of the masked data to further determine the real length of original data. Here, the DR targets to obtain the final sign indicator f from $[m_1]$.

Step 3 (Data Preparation @ DSP). The DSP chooses three random numbers R ($\mathcal{L}(R) < \mathcal{L}(n)/4$), c_1 and ck_1 . It first encrypts "1" and then computes as follows:

- 1) $[1] = \{(1+n) * PK^{r'}, g^{r'}\}$;
- 2) $[2*m_1 + 1] = (T, T') = [m_1]^2 * [1] = \{(1 + (2*m_1 + 1) * n) * PK^{r'+2*r_1}, g^{r'+2*r_1}\}$;
- 3) Then it flips a coin s . If $s = -1$; it computes:

$$[m']_{pk_{CP}} = \{T^{n-R}, (T')^{a*(n-R)}\} = [-R * (2*m_1 + 1)].$$

- 4) Otherwise ($s = 1$), it calls *PDec1* and computes: $[m']_{pk_{CP}} = \{T^R, T'^{a*R}\} = [R * (2 * m_1 + 1)]$.
- 5) The DSP computes $c_2 = (ck_1)^{-1} \bmod n$, and $s' = c_1 * c_2 * s \bmod n$.

The data packet sent to the CP is $\{[m']_{pk_{CP}}, s'\}$.

Step 4 (Data Process @ CP). Upon receiving the data packet from the DSP, the CP decrypts $[m']_{pk_{CP}}$ with *PDec2* to obtain raw data m' .

The CP compares $\mathcal{L}(m')$ with $\mathcal{L}(n)/2$. If $\mathcal{L}(m') < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$.

The CP chooses a random number r and a second partial key ck_2 , and further computes as follows:

- 1) $[\hat{f}]_{pk_{ck_2}} = (\bar{T}, \bar{T}') = \{(1+s'u*n)g^{ck_2*r}, g^r\}$;
- 2) Encrypt ck_2 using ABE: $CK'_2 = Enc^{ABE}(ck_2, \gamma, PK')$.

Finally, the CP forwards $[\hat{f}]_{pk_{ck_2}}$ to DSP.

Step 5 (Additional Process @ DSP). The DSP further processes the data packet as follows:

- 1) Compute $c_3 = c_1^{-1} \bmod n$;
- 2) $[f]_{pk_{ck}} = \{\bar{T}^{ck_1 * c_3}, (\bar{T})^{c_3}\} = \{(1 + su * n)g^{ck_1 * ck_2 * r * c_3}, g^{r * c_3}\}$.
- 3) $CK' = Enc^{ABE}(ck_1, \gamma, PK') * CK'_2$.

Step 6 (Data Access @ DR). The DR satisfying the access policy in ABE can decrypt CK' to obtain ck and further decrypt $[f]_{pk_{ck}}$ to obtain f .

Note: if $f = 1, m_1 \geq 0$; Otherwise, $m_1 < 0$.

4.4.5 Absolute

Besides the operations in *Sign Acquisition*, *Absolute* needs to mask the original data by raising to the power of a random number, and then remove the mask according to sign indicator to achieve a final absolute result. We assume that $\mathcal{L}(m) < \mathcal{L}(n)/4$ and that BIG is the largest raw data of m . Then the raw data is in the scope $[-BIG, BIG]$. Here, given ciphertext $[m_1]$, DR wants to get the absolute value $|m_1|$.

Step 3 (Data Preparation @ DSP). The DSP chooses three random numbers R where $\mathcal{L}(R) < \mathcal{L}(n)/4$, c_1 and c_2 , and also chooses the first partial key ck_1 . It first encrypts "1" and then computes as follows:

- 1) $[1] = \{(1 + n) * PK^{r'}, g^{r'}\}$;
- 2) $[2 * m_1 + 1] = (T, T') = [m_1]^2 * [1] = \{(1 + (2 * m_1 + 1) * n) * PK^{r' + 2 * r_1}, g^{r' + 2 * r_1}\}$;
- 3) Then it flips a coin s . If $s = -1$; it computes:

$$[m']_{pk_{CP}} = \{T^{n-R}, (T')^{a * (n-R)}\} = [-R * (2 * m_1 + 1)]$$

Otherwise ($s = 1$), it calls *PDec1* and computes:

$$[m']_{pk_{CP}} = \{T^R, T^{a * R}\} = [R * (2 * m_1 + 1)];$$

- 4) Compute $[c_1 m_1] = [m_1]^{c_1}$, and call *PDec1* to obtain $[c_1 m_1]_{pk_{CP}}$.
- 5) The DSP sets $c_3 = (ck_1)^{-1} \bmod n$, and $s' = c_2 * c_3 * s \bmod n$.

The data packet sent to the CP is $\{[m']_{pk_{CP}}, s', [c_1 m_1]_{pk_{CP}}\}$.

Step 4 (Data Process @ CP). Upon receiving the data packet from the DSP, the CP decrypts $[c_1 m_1]_{pk_{CP}}$ and $[m']_{pk_{CP}}$ with *PDec2* to obtain $c_1 m_1$ and m' , respectively. The CP compares $\mathcal{L}(m')$ with $\mathcal{L}(n)/2$. If $\mathcal{L}(m') < \mathcal{L}(n)/2$, it sets $u = 1$; otherwise, $u = -1$.

Then CP chooses r and the second partial key ck_2 , and further computes as follows:

- 1) $[c_1 m_1 s' u]_{pk_{ck_2}} = (\bar{T}, \bar{T}') = \{(1 + c_1 m_1 s' u * n)g^{ck_2 * r}, g^r\}$;
- 2) Encrypt ck_2 using ABE: $CK'_2 = Enc^{ABE}(ck_2, \gamma, PK')$.

Finally, the CP forwards $[c_1 m_1 s' u]_{pk_{ck_2}}$ and CK'_2 to DSP.

Step 5 (Additional Process @ DSP). The DSP further processes the data packet as follows:

- 1) Set $c_4 = (c_1)^{-1} \bmod n$ and $c_5 = (c_2)^{-1} \bmod n$;
- 2) $[su * m_1]_{pk_{ck}} = \{\bar{T}^{ck_1 * c_4 * c_5}, \bar{T}^{r * c_4 * c_5}\} = \{(1 + su * m_1 * n)g^{ck_1 * ck_2 * r * c_4 * c_5}, g^{r * c_4 * c_5}\}$;
- 3) $CK' = Enc^{ABE}(ck_1, \gamma, PK') * CK'_2$.

Step 6 (Data Access @ DR). The DR that satisfies the access policy in ABE can decrypt CK' to obtain ck . The DSP sends

the data packet $[su * m_1]_{pk_{ck}}$ to the DR in a secure way. Then the DR can decrypt it to obtain $su * m_1$.

Note: if $m_1 \geq 0, su = 1$; Otherwise, $su = -1$. Hence, $su * m$ is the absolute of data m .

4.4.6 Comparison

Comparison can be simply accomplished by checking the sign of the difference value of two data by calling *Sign Acquisition*. Similar to the functions above, DR wants to compare the raw data (m_1, m_2) based on their encrypted data. For ease of presentation, $m_1 - m_2$ is denoted as m_{1-2} .

$$[m_1] = (T_1, T_1') = \{(1 + m_1 * n) * PK^{r_1}, g^{r_1}\}$$

$$[m_2] = (T_2, T_2') = \{(1 + m_2 * n) * PK^{r_2}, g^{r_2}\}$$

Step 3 (Data Preparation @ DSP). DSP first computes to get the subtraction of encrypted data:

$$(T, T') = \{T_1 * (T_2)^{n-1}, T_1' * (T_2')^{n-1}\} = [(m_1 - m_2)].$$

The following steps are the same as those in *Sign Acquisition*, which are skipped for the reason of paper length limitation. Through the cooperation of the DSP and the CP, the DR finally gets the sign of $m_{1-2} = m_1 - m_2$. In the end, the DR can obtain the comparison result. If $m_{1-2} \geq 0, m_1 \geq m_2$; otherwise, $m_1 < m_2$.

4.4.7 Equality Test

Equality Test needs to check the signs of both difference value and negative difference value of original two data by calling *Comparison* twice. DR wants to know whether m_1 is equal to m_2 or not from the encrypted data $([m_1], [m_2])$. The DSP and the CP directly interact with each other in two parallel computations of *Comparison*.

They compare m_1 and m_2 in two forms: 1) $m_{1-2} = m_1 - m_2$; 2) $m_{2-1} = m_2 - m_1$. Through the operations in *Comparison*, DSP can get two results $[f_1]_{pk_{ck}}$ and $[f_2]_{pk_{ck}}$ respectively. Then the DSP can obtain $[f]_{pk_{ck}} = [f_1 + f_2]_{pk_{ck}} = [f_1]_{pk_{ck}} * [f_2]_{pk_{ck}}$.

Finally, the DR that satisfies the access policy in ABE can decrypt CK' to obtain ck . The DSP sends the data packet $[f]_{pk_{ck}}$ to the DR in a secure way. Then the DR can further decrypt $[f]_{pk_{ck}}$ to obtain f .

Note: if $f = 2, m_1 = m_2$; Otherwise, $m_1 \neq m_2$.

4.5 Data Analysis over Fixed Point Numbers

In order to adapt to various applications, we further design a scheme to deal with fixed point numbers rather than integers.

Normally, a fixed-point number can be represented with three parts: the sign field, integer field and fractional field. Given a fixed point number m , we set it in binary format as $(S, b_{e-2}, b_{e-3}, \dots, b_0, b_{-1}, \dots, b_{-f})$, where S is its sign, $e - 1$ is the length of the integer field, f is the length of the fractional field and its value is $m = S * \sum_{i=-f}^{e-2} (b_i * 2^i)$. In order to encrypt number m , we should first scale $|m|$ to $|m'| = |m| * 2^f$. If m is a positive value, then we can set $m' = m * 2^f$; otherwise, we should set $m' = N - m * 2^f$. Then m' as an integer can be encrypted as $[m']$.

Regarding to the operations presented in Section 4.4 except multiplication, it is easy to be executed over fixed point numbers.

4.5.1 Multiplication over Encrypted Fixed Point Numbers

For two fixed point numbers m_1 and m_2 , we perform as follows:

- 1) Get the integer representation of m_1 and m_2 : $m'_1 = m_1 * 2^f$ and $m'_2 = m_2 * 2^f$;
- 2) Users upload their ciphertext to CSP as: $[m'_1]$ and $[m'_2]$.
- 3) Then execute Multiplication over $[m'_1]$ and $[m'_2]$;
- 4) Finally, DR can get the value of $m'_1 * m'_2$, then scale down to obtain the final result $m_1 * m_2 = m'_1 * m'_2 * 2^{-f}$.

4.5.2 Polynomial Computations over Encrypted Fixed Point Numbers

For a clear presentation, we give an example of $m_1 * m_2 + m_3$. The detailed procedure is presented as follows:

- 1) (@DPs) Get the integer representation of three numbers: $m'_1 = m_1 * 2^f$, $m'_2 = m_2 * 2^f$ and $m'_3 = m_3 * 2^f$;
- 2) (@DSP) Execute the same operations to the Step 3 in Multiplication to get: $\{[c_1 * m'_1]_{pk_{CP}}, [c_2 * m'_2]_{pk_{CP}}\}$.
- 3) (@CP) Decrypt data packet to get $c_1 * m'_1$ and $c_2 * m'_2$; Then encrypt $c_1 * m'_1 * c_2 * m'_2$ with Diffie-Hellman key PK to get $[c_1 c_2 * m'_1 m'_2]$; Send it to DSP.
- 4) (@DSP) First compute $[m'_3]^{2^f}$ to scale the original data and get $[2^f * m'_3]$; Execute the proposed scheme Addition over $[2^f * m'_3]$ and $[c_1 c_2 * m'_1 m'_2]$ under the cooperation with CP.

Note: The length of data encrypted via Paillier system is less than $\mathcal{L}(n)$. In order to support various computations above over positive and negative numbers, we strictly restrict the length of data such that $\mathcal{L}(m) < 1/2 * \mathcal{L}(n)$. If multiplication of N pieces of data is needed, then it has a strict limitation that $\mathcal{L}(m) = (e + f) < \mathcal{L}(n)/(2N)$. Here, with the default setting $\mathcal{L}(n) = 1024$, $e = 45$, $f = 15$, we should limit the multiplication times (i.e., the number of provided data) to 8.

5 SECURITY ANALYSIS AND PERFORMANCE EVALUATION

5.1 Security Analysis

The semantic security of HRES has been proved in our previous work [18]. Hence, we skip its security proof and focus on the security analysis of our proposed schemes. Here, we adopt the security model for securely realizing an ideal functionality in the presence of semi-honest (non-colluding) adversaries. It involves four types of parties: DSP, CP, DP and DR. Thus, we construct four simulators $Sim = (Sim_{DP}, Sim_{DSP}, Sim_{CP}, Sim_{DR})$ to against four kinds of adversaries $(\mathcal{A}_{DP}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DR})$ that corrupt DP, DSP, CP and DR , respectively.

Theorem 1. *The Addition scheme in Section 4.4.1 can securely obtain the plaintext of addition via computations on ciphertexts in the presence of semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{DR}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DP})$.*

Proof. We present the construction of four independent simulators $(Sim_{DP}, Sim_{DSP}, Sim_{CP}, Sim_{DR})$. Here, we prove the security of the case with two inputs (i.e., $N = 2$). \square

Sim_{DP} receives the input of m_1 and m_2 , then it simulates \mathcal{A}_{DP} as follows: it encrypts data m_1 as $[m_1] = EncTK(m_1)$, and data m_2 as $[m_2] = EncTK(m_2)$. Finally, it returns $[m_1]$ and $[m_2]$ to \mathcal{A}_{DP} , and outputs the entire view of \mathcal{A}_{DP} .

The view of \mathcal{A}_{DP} is the encrypted data. The views of \mathcal{A}_{DP} in the real and the ideal executions are indistinguishable.

Sim_{DSP} simulates \mathcal{A}_{DSP} as follows: it runs the $EncTK$ on two randomly chosen numbers \tilde{m}_1 and \tilde{m}_2 ; then it multiplies $[\tilde{m}_1]$ by $[\tilde{m}_2]$ to get $[\tilde{m}]$ where $m = \tilde{m}_1 + \tilde{m}_2$; further it masks the ciphertext with two random numbers r_1 and ck_1 , and runs the $PDec1$ to obtain $[c_1(\tilde{m} + r_1)]_{pk_{CP}}$. By accessing Sim_{CP} , it receives $[\tilde{m}]_{pk_{ck_2}}$ and \widetilde{CK}'_2 based on a randomly generated key ck_2 . Then it computes to obtain $[\tilde{m}]_{pk_{ck}}$ and \widetilde{CK}' . Finally, Sim_{DSP} outputs $[c_1(\tilde{m} + r_1)]_{pk_{CP}}$, $[\tilde{m}]_{pk_{ck_2}}$, \widetilde{CK}'_2 , $[\tilde{m}]_{pk_{ck}}$ and \widetilde{CK}' to \mathcal{A}_{DSP} . If \mathcal{A}_{DSP} replies with \perp , Sim_{DSP} returns \perp .

The view of \mathcal{A}_{DSP} consists of the encrypted data and the ciphertext of corresponding decryption key. Owing to the honesty of the challenged cloud user and the security of the HRES, \mathcal{A}_{DSP} receives the same output in both real and ideal executions. As the decryption key ck is randomly constructed by CSP and CP in each challenge, \mathcal{A}_{DSP} would not obtain more information by executing multiple challenges. Thus, the views are indistinguishable.

Sim_{CP} simulates \mathcal{A}_{CP} as follows: it randomly chooses data \tilde{m} , uses Enc to obtain $[\tilde{m}]_{pk_{ck_2}}$ and calls ABE encryption to obtain \widetilde{CK}'_2 . Then it sends $[\tilde{m}]_{pk_{ck_2}}$ and \widetilde{CK}'_2 to \mathcal{A}_{CP} . If \mathcal{A}_{CP} replies with \perp , Sim_{CP} returns \perp . In both real and ideal executions, it receives the output of two ciphertexts. In the real world, the security is guaranteed by the semantic security of HRES and the security of ABE.

Sim_{DR} simulates \mathcal{A}_{DR} as follows: (it cannot enquire for the challenged data) it randomly chooses data $[m']_{pk_{ck}}$ and decrypts it to obtain m' , and then sends it to \mathcal{A}_{DP} . If \mathcal{A}_{DP} replies with \perp , Sim_{DR} returns \perp .

The view of \mathcal{A}_{DR} is the decrypted result without any other information. But in both the real and ideal executions, it is guaranteed by the semantic security of the HRES. The views are indistinguishable in both executions.

No matter how many times the adversary accesses the simulator \mathcal{A}_{DR} , it is still difficult to obtain the original real data for two reasons: 1) the randomly chosen data have no relation to the original real data; 2) exhaustion attack is hard owing to the randomness of the chosen numbers.

The security proofs of other operations are similar to that of the Addition under the semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{DR}, \mathcal{A}_{DSP}, \mathcal{A}_{CP}, \mathcal{A}_{DP})$.

5.2 Performance Evaluation

In this section, we analyze the computational complexity and the communication overhead of our proposed seven computing operation schemes. Further, we implemented them and tested their performances through simulations.

5.2.1 Computational Complexity Analysis

Herein, we analyze the computational complexity of our designed schemes. Notably, the ABE and the HRES are based on different cryptographic techniques, while the computational complexity by employing ABE to control access in each operation is the same. Hence, we analyze the computational complexity of ABE in a separate part.

1) *The Computational Complexity of ABE.* As presented in Section 4.1.3, the computations of all four algorithms are related to the number of attributes. In order to clarify the relationships, we assume that there are $|U|$ universal attributes, that $|\gamma|$ attributes are in the access policy tree \mathcal{T} , and that at most ϑ attributes should be satisfied in the policy tree \mathcal{T} to decrypt ciphertext.

The setup algorithm $Setup^{ABE}()$ should choose the system parameters and generate the public parameters. It needs to do $|U| + 1$ exponentiations and one bilinear pairing. The encryption algorithm $Enc^{ABE}()$ needs to generate the ciphertext by involving each attribute in \mathcal{T} with one exponentiation, and encrypt the original message with one exponentiation. Hence, the encryption algorithm needs $|\gamma| + 1$ exponentiations in total. $KeyGen^{ABE}()$ operates $|\gamma| + 1$ exponentiations to generate decryption key. $HE^{ABE}()$ merely involves $|\gamma| + 1$ multiplications. As exponentiation is significantly more costly than multiplication, we ignore the computation cost of $HE^{ABE}()$ in our analysis. The main operation in $Dec^{ABE}()$ is to compute the divided pieces of encrypted partial keys, which performs at most ϑ bilinear pairings.

The system setup and data encryption only need to be executed once. Thus, their computational complexity can be amortized when multiple users access the final result. Though the computation cost of cloud users in decryption algorithm is higher than our previous scheme [18], it enhances the security of processing results by introducing fine-grained access control. The system should balance efficiency and security according to its capabilities and requirements. Moreover, trust can be used as a single attribute in ABE in order to greatly reduce computational cost [15], [16], [17].

2) *The Computational Complexity of Data Analysis.* As the computations related to ABE for access control neither vary with the number of provided data nor vary with the designed functions, their costs are ignored in the following analysis.

The modular exponentiation operation is significantly more time-consuming than the modular addition and multiplication, thus we ignore the fixed numbers of additions and multiplications in the following analysis. In addition, the computational complexity of basic operations (modular exponentiation in HRES, exponentiation and bilinear pairing in ABE) would never be affected by the number of provided data or access policy. Thus, we set all their computational complexities to be $\mathcal{O}(1)$.

Enc and $EncTK$ need two modular exponentiations. Both the algorithms Dec and $PDec2$ take one modular exponentiation and one modular multiplication to obtain the ciphertext, and $PDec1$ needs one modular exponentiation. Based on these analyzed results, we further give the computational complexity in seven operation functions. We hold the assumption that there are N pieces of provided data in *Addition*, *Subtraction*, and *Multiplication*.

Computational Complexity of DP. The DP directly outsources its data with $EncTK$ for data processing/analyzing. Its computation cost of outsourcing one piece of data is two modular exponentiations, which is with the computational complexity $\mathcal{O}(1)$.

Computational Complexity of DSP. In *Addition*, the DSP has two parts of computations in each algorithm. In Step 3, the DSP first needs to multiply the N pieces of data, mask ciphertexts with random numbers through two modular

exponentiations, and then call $PDec1$. Moreover, it should do one modular exponentiation for removing the mask in Step 5. In *Subtraction*, the DSP needs to obtain the negative of subtractor with two more modular exponentiations than that in *Addition*. As the modular exponentiation of g^r with $r \in [1, n]$ needs at most n modular multiplications, the multiplication of N ($N < n$) pieces of provided data results in computational complexity of $\mathcal{O}(1)$. Thus, the DSP has the computational complexity of $\mathcal{O}(1)$ in both *Addition* and *Subtraction*.

In *Multiplication*, the DSP should first mask each piece of data with a random number through two exponentiations and then call $PDec1$ to partially decrypt the data in Step 3, which results in $3N$ modular exponentiations. In Step 5, it should do one exponentiation to remove the mask from original data. Hence, the computational complexity of DSP results in $\mathcal{O}(N)$.

In *Sign Acquisition*, the DSP needs to do four modular exponentiations in Step 3, and two more in Step 5. In *Absolute*, the DSP needs to do three more modular exponentiations in Step 3 than it does in *Sign Acquisition*. In *Comparison*, the DSP should first get the subtraction of two pieces of encrypted data, which needs two modular exponentiations, and then do the same operation as it does in *Sign Acquisition*. In *Equality Test*, the DSP needs invoke the *Comparison* twice. Hence, its computational complexity in these three schemes are all $\mathcal{O}(1)$.

Computational Complexity of CP. In the schemes except *Multiplication*, *Absolute* and *Equality Test*, the CP first decrypts ciphertext by calling $PDec2$ and then encrypts the data with a partial secret key, which needs three modular exponentiations in total. Hence, the computational complexities in them are all $\mathcal{O}(1)$ regardless of the ABE encryption.

In *Multiplication*, the CP should first decrypt each ciphertext of masked original data with $PDec2$, then encrypt their product with Enc . It totally needs $(N + 2)$ modular exponentiation and results in computational complexity $\mathcal{O}(N + 2)$.

Different from the analysis above, the CP in *Absolute* should first do two partial decryptions $PDec2$. Generally, it needs four modular exponentiations. Its computational complexity results in $\mathcal{O}(1)$.

In *Equality Test*, the CP takes the double computation costs in *Comparison*, which needs six modular exponentiations but still results in the computational complexity of $\mathcal{O}(1)$.

Computational Complexity of DR. The DR should first call $Dec^{ABE}()$ to obtain the decryption key ck . Then, it further decrypts the pre-processing result to obtain the final result by calling Dec , which needs one modular exponentiation.

3) *The Summary of Computation Analysis in Seven Operation Schemes.* In the following tables and figures, we use the following corresponding notations: *Addition* (Add), *Subtraction* (Sub.), *Multiplication* (Mul.), *Sign Acquisition* (Sign), *Absolute* (Abs.), *Comparison* (Comp.), and *Equality Test* (Equal). Here, we present the total computation costs in each algorithm and compare them with existing work [18] in Table 2. The constant number of multiplications is not considered in the table. We can observe that our new schemes introduce some computation cost to the involved entities, but it can greatly reduce the cost of DSP and CP when the number of DRs is high, especially in *Multiplication*, which is further discussed in Section 5.2.2. Moreover, our scheme provides fine-grained access control and enhances the security of processing results.

TABLE 2
Computation Analysis

Role	Operation	Computations of our work	Complexity of our work	Computations in [18]
DP	ALL	$2 * ModExp$	$\mathcal{O}(1)$	$2 * ModExp$
	Add	$N * ModMul + 4 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$(N * ModMul + 3ModExp)N_R$
	Sub.	$N * ModMul + 6 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$(N * ModMul + 5ModExp)N_R$
	Mul.	$(3N + 1)ModExp + (\gamma + 1)Exp$	$\mathcal{O}(N + \gamma)$	$N_R(2N + 2) * ModExp$
DSP	Sign	$6 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$3N_R * ModExp$
	Abs.	$9 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	–
	Comp.	$8 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$4N_R * ModExp$
	Equal	$16 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$8N_R * ModExp$
	Vari.	–	–	$N_R(4N + 2) * ModExp$
	Add	$3 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$3N_R * ModExp$
	Sub.	$3 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$3N_R * ModExp$
	Mul.	$(N + 2)ModExp + (\gamma + 1)Exp$	$\mathcal{O}(N + \gamma)$	$N_R(N + 2) * ModExp$
	Sign	$3 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$3N_R * ModExp$
CP	Abs.	$4 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	–
	Comp.	$3 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$3N_R * ModExp$
	Equal	$6 * ModExp + (\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	$6N_R * ModExp$
	Vari.	–	–	$N_R((N + 2) * ModExp + (\gamma + 1)Exp)$
Authority	ALL	$N_R(\gamma + 1)Exp$	$\mathcal{O}(\gamma)$	–
DR	ALL	$1 * ModExp + \vartheta * BiPair$	$\mathcal{O}(\vartheta)$	$1 * ModExp$

Notes: N : the number of provided data; $BiPair$: the bilinear pairing in ABE; Exp : the exponentiation in ABE; $ModExp$: the modular exponentiation in HRES; $ModMul$: the modular multiplication; ALL: fits all schemes; $|\gamma|$: the number of attributes in access policy tree T ; ϑ : the number of attributes needed to satisfy policy T ; N_R : the number of data requesters.

5.2.2 Communication Overhead

Each ciphertext is composed of two parts: $[m_i] = \{T_i, T_i'\}$. It is highly related to the length of n^2 , which has $2\mathcal{L}(n)$ bits. Hence, it has to transmit $4\mathcal{L}(n)$ bits for each ciphertext. Here, we set the bit length of each element in ABE to be \mathcal{L}_Δ .

Further, we summarize the communication overhead in each scheme and compare with existing work with the assumption of N pieces of provided data, which is shown in Table 3. Notably, the communication overhead in Data Analysis (e.g., Add., Sub. and Mul., etc.) can be amortized over multi-user accesses, while the communication cost during Data Access is needed for each authorized DR. Moreover, the communication overhead caused by data provision can also be amortized by other data processing, which is not counted in Table 3. We can observe that the communication overhead in *Addition* and *Subtraction* does not rely on the number of provided data, while the

TABLE 3
Communication Overhead of Each Scheme

Schemes	Communication overhead of our work	Communication Overhead in [18]
Add	$8\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta$	$8\mathcal{L}(n)N_R$
Sub.	$8\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta$	$8\mathcal{L}(n)N_R$
Mul.	$4(N + 1)\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta$	$4(N + 3)\mathcal{L}(n)N_R$
Sign	$9\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta$	$12\mathcal{L}(n)N_R$
Abs.	$13\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta$	–
Comp.	$9\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta$	$12\mathcal{L}(n)N_R$
Equal	$18\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta$	$24\mathcal{L}(n)N_R$
Vari.	–	$4(2N + 1)\mathcal{L}(n)N_R$
Data Access	$N_R(4\mathcal{L}(n) + (\gamma + 1)\mathcal{L}_\Delta)$	$4\mathcal{L}(n)N_R$

Note: \mathcal{L}_Δ : bit length of elements in ABE; N_R : the number of data requesters.

communication overhead in *Multiplication* is proportional to the number of provided data. But the cost in other schemes does not vary much as it has only one or two data inputs. From the comparisons, we can find that the communication cost during data analysis in our new scheme is independent of the number of DRs, which saves the cost for multi-user access. In general, the communication cost is reasonable and our schemes are suitable for various applications.

5.2.3 Experimental Results

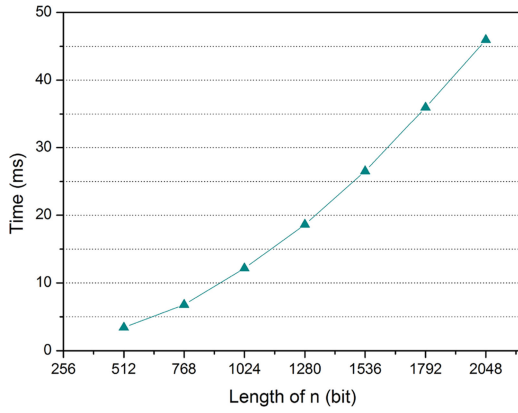
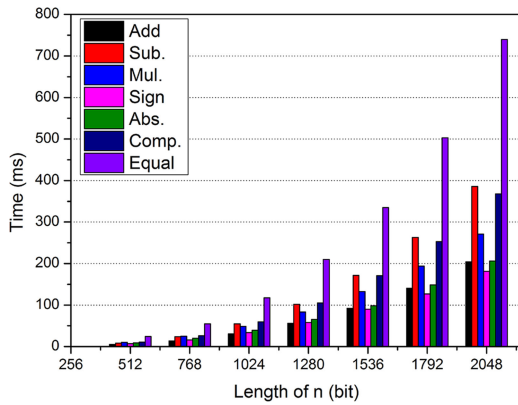
The efficiency of HRES has been presented in [18]. Hence, we focused on the performance evaluation of the proposed seven operation schemes. We implemented the proposed seven computing operation functions and tested their performances to check with aforementioned theoretical analysis. The evaluations are performed on a laptop with Intel Core i5-3337U CPU 1.8 GHz and 8 GB RAM with Java Pairing-Based Cryptography library (jPBC). To achieve better accuracy, we tested each algorithm 500 times and reported the average value of all testing results. Unless particularly specified, the parameters in our tests are set as the default values listed in Table 4.

Note: In our simulations, we set $\mathcal{L}(ck_1) = \mathcal{L}(ck_2) = 250$ bits and employ the curve of TYPE A in jPBC (<http://gas.dia.unisa.it/projects/jpbc/docs/ecpg.html#TypeA>). If higher security is desired, TYPE E can be adopted to support keys with longer length $\mathcal{L}(ck_1) = \mathcal{L}(ck_2) = 500$ bits. In our test machine, one pairing can be computed in approximately 9.8 milliseconds (ms).

In our proposed schemes, we adopt HRES to achieve the privacy-preserving data processing, while ABE is used to realize the data access control. As they are influenced by different parameters, we analyze the efficiency of data processing and access control separately. The CP in Step 4 and DSP in Step 5 encrypt the partial decryption key using ABE and

TABLE 4
Parameter Settings

Parameter	Value (bits)
$\mathcal{L}(n)$	1024
$\mathcal{L}(m_i)$	$\mathcal{L}(n)/4$
$\mathcal{L}(sk_i) = \mathcal{L}(sk_j) = \mathcal{L}(sk_{DSP}) = \mathcal{L}(sk_{CP})$	500
$\mathcal{L}(ck_1) = \mathcal{L}(ck_2)$	250
$\mathcal{L}(r_*) = \mathcal{L}(c_*)$	500
$\mathcal{L}(R)$	$(\mathcal{L}(n)/4) - 10$

Fig. 3. Operation time of DPs with different length of n .Fig. 4. Operation time of DSP in Step 3 with different length of n .

the DR decrypts to obtain the raw key with ABE in Step 6, the computation time of which are all presented in *Test 4*. Other data processing computation costs introduced by the ABE algorithm are simulated in *Tests 1, 2* and *3*.

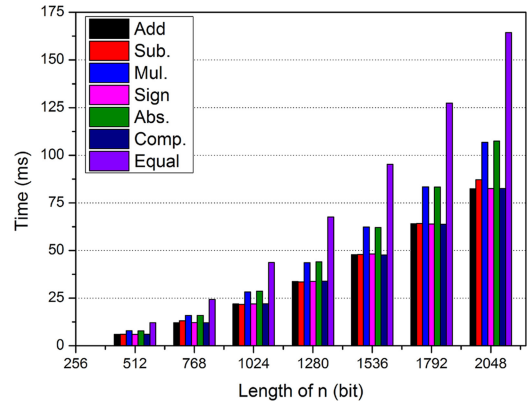
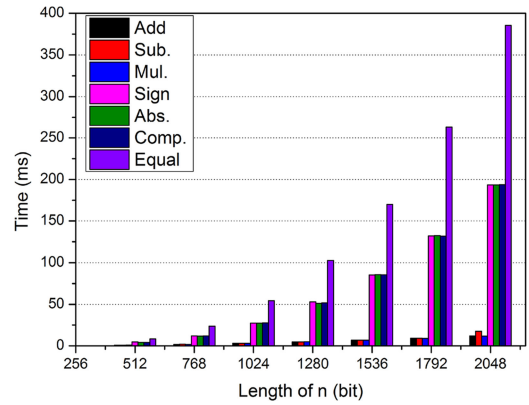
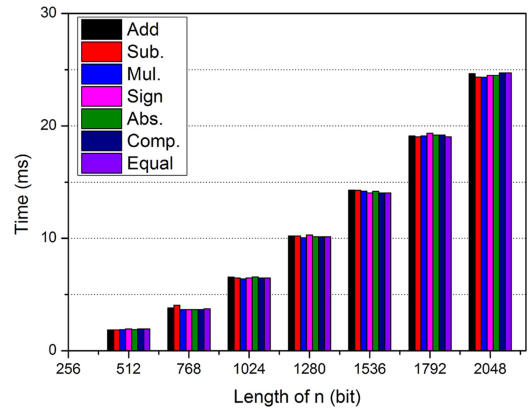
1) Efficiency of Data Processing

Test 1: Influence of the length of n on data processing

We presented the computation time of four involved entities: DP, DSP, CP, and DR respectively. The DP only involves in Step 2, which is similar in all schemes. We directly tested its computation cost with different length of n as shown in Fig. 3. We can observe that it is efficient and acceptable for DP with constrained resources.

The computation costs of DSP include two parts: Step 3 and Step 5. For a clearer presentation, we do not combine them, but present them in Figs. 4 and 6 respectively. We can observe that the cost grows with the increase of the bit length of n . In addition, the *Equality Test* is the most time-consuming, which needs about 750 ms when n achieves 2048 bits, while other algorithms take less than 400 ms. The DSP needs to process the *Comparing* twice to complete *Equality Test*. The performance evaluation of which is consistent with our analysis in Section 5.2.2. The computation cost of the DSP in Step 5 is shown in Fig. 6. Generally, our proposed schemes are acceptable for the cloud service provider, DSP.

From Fig. 5 that shows the computation cost of the CP, the *Equality Test* is also the most time-consuming while others take merely about 100 ms. As shown in Fig. 7, the computation cost of the DR is much less than those of two servers. Even when the bit length of n reaches 2048 bits, it still only needs about 25 ms to complete the computation.

Fig. 5. Operation time of CP with different length of n .Fig. 6. Operation time of DSP in Step 5 with different length of n .Fig. 7. Operation time of DR with different length of n .

In general, the above tests prove that most computation costs are undertaken by the DSP and the CP. The cloud users do not have much computation overhead. This result shows the practical advantage of the proposed schemes in the usage of mobile environments.

2) Flexibility of Addition, Subtraction and Multiplication

The operations of DSP include Step 3 and Step 5, the time of which are marked separately as DSP(1) and DSP(2) in Figs. 8, 9, and 10.

Test 2: Performance of Addition and Subtraction with a large number of provided data

In this experiment, we tested the performance of three kinds of system entities (i.e., DSP, CP and DR) in *Addition* with different numbers of provided data ($N = 10, 10^2, 10^3, 10^4, 10^5$). As shown in Fig. 8, the computation cost of the

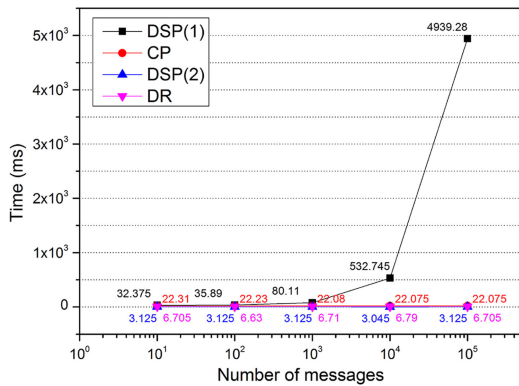


Fig. 8. Operation time of each entity in *Addition* with different number of DPs.

DSP in Step 3 increases with the number of the provided data, while it does not influence other steps. Even when the number of provided data reaches 10^4 , the DSP only takes about 530 ms. It takes almost the same time for the DR to complete the computation with different numbers of provided pieces of data. This fact implies that our schemes can deal with a great number of data for addition efficiently.

We assume that the subtraction formula is $(\sum_{i=1}^W m_i - \sum_{i=W+1}^N m_i)$ with $W = N/2$, which means that half of the provided data is subtracted from the sum of another half. Fig. 9 shows its performance, which is similar to that of *Addition* and is efficient to support a big number of DPs.

In general, the scheme can flexibly be used in various situations with different number of data providers.

Test 3: Performance of Multiplication with a large number of provided data

Different from *Addition* and *Subtraction*, the *Multiplication* is time-consuming and communication-consuming. It is not flexible and possible to support the computation of a huge number of provided data. Thus, we only tested it with limited numbers ($N = 100, 200, 300, 400, 500, 600, 700, 800$). Moreover, the original data length is set as $\mathcal{L}(n)/N$.

As shown in Fig. 10, it takes about 15 seconds to finish the computation in Step 3 for the DSP when the number of provided messages achieves 800. However, the data numbers do not influence the computation cost of the DR and no extra overhead is introduced, which merely costs about 6.7 ms.

3) Efficiency of Attribute-Based Encryption

The non-colluding servers and the DRs employ ABE to realize the flexible access control over the result of data

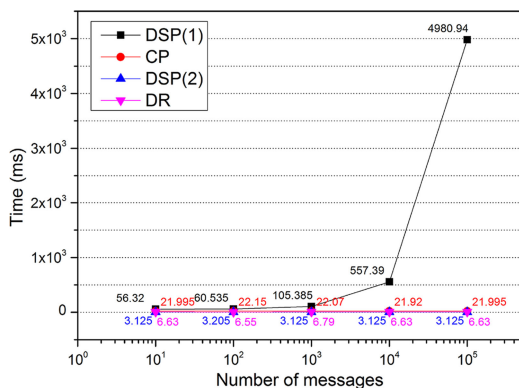


Fig. 9. Operation time of each entity in *Subtraction* with different number of DPs.

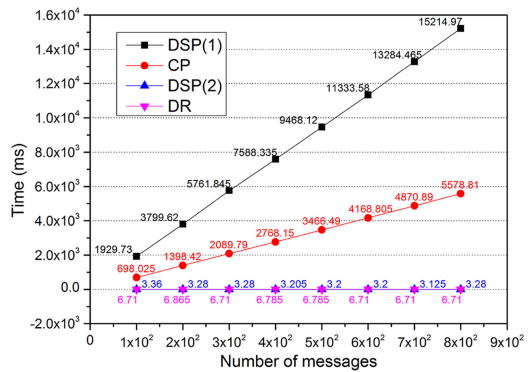


Fig. 10. Operation time of each entity in *Multiplication* with different number of DPs.

processing and the data retrieve, respectively. Here, we focus on the performance of ABE through one test.

Test 4: Performance of ABE with different numbers of attributes

In this experiment, we tested the performance of five steps of ABE with different number of attributes: System Setup (Setup_ABE), Encryption (Enc_ABE), Key Generation (KeyGen_ABE), Decryption (Dec_ABE) and its homomorphic computation (HE_ABE). The test was executed with the setting that all universal attributes are involved in encryption and that one attribute is needed to satisfy the policy tree. That is, $\mathcal{O}(|U|) = \vartheta$, which varies from 2 to 8 in our tests while $\mathcal{O}(|\gamma|) = 1$.

Fig. 11 shows the costs of all operations in ABE except the algorithm HE_ABE, as it only takes less than 1ms. By applying trust-based access control schemes [15], [16], [17], the decryption only involves one attribute and takes only 10 ms. We can observe that the computation cost of other algorithms is proportional to the number of attributes. Though employing ABE would cause high computation overhead, high security and fine-grained access control can be supported. Most computations are taken by cloud servers. In addition, the cloud servers only need to perform the setup and encryption once, which can be amortized by multiple accesses of DRs.

4) Experimental Comparison with Existing Work

Test 5: Performance comparison of Multiplication schemes in our work with an existing scheme in [18]

In this test, we compared the performance of *Multiplication* in our scheme with an existing scheme [18], as shown in Fig. 12. We obtained the total computation time of all

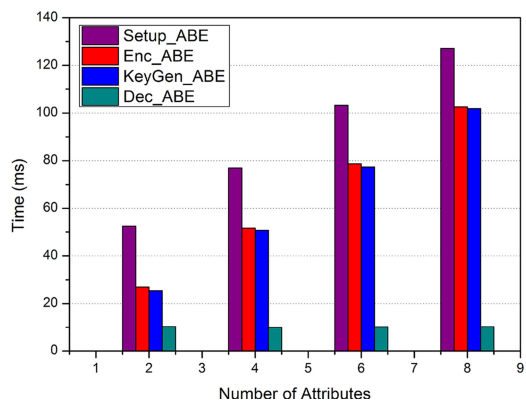


Fig. 11. Operation time of KPABE with different numbers of attributes.

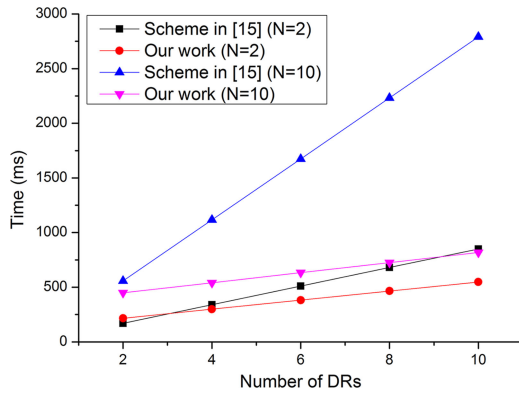


Fig. 12. Performance comparison of *Multiplication* schemes in our work and previous scheme [18].

involved entities with different number of DRs ($N_R = 2, 4, 6, 8, 10$). By comparing the performance of the two schemes with two DPs, we can see that our proposed scheme outperforms that in [18] when the number of DRs is over 3. Moreover, we compared the performance of two schemes with different number of DPs ($N = 2, 10$). By observing the simulation results, we find that our scheme shows much superiority in performance, especially for a big number of DPs.

5.3 Comparison with Existing work

Herein, we further compare our proposed scheme with some related work. With regard to secure data processing, HE-based schemes [5], [30] provide a feasible way to deal with the computation over encrypted data. But they have a serious problem because they are all single-user systems, which restricts the possibility of multi-user access control. SMC-based scheme [27] enables flexible access via secret sharing. However, they all introduce high computation overhead in big data processing, especially for multiplication. For a more intuitive comparison, we compare their communication cost and computation cost in multiplication over N pieces of data with our scheme in Table 5. The result shows the advantages of our scheme in terms of communication and computation efficiency. Moreover, no existing work overcomes the challenges to realize multiple computations over encrypted data and flexible access control over the processing result. Obviously, our work overcomes the current research challenges and effectively achieves flexible and secure computations over encrypted data with fine-grained access control.

6 CONCLUSION

In this paper, we proposed an efficient and secure scheme to achieve privacy-preserving data processing with ABE-based flexible access control. It can support seven basic operations and achieve fine-grained access control without the need of fully trusted cloud servers. Security analysis, performance evaluation and performance comparison with existing work further demonstrated that our scheme is efficient and effective with regard to big data processing operations. In the future, we are going to realize more operations, improve scheme efficiency and overcome latency by applying edge computing and pre-processing technologies. On the other hand, we will further explore significant applications of our scheme towards practical use.

TABLE 5
Comparison with Existing Work in Multiplication

	Communication cost (bits)	Computation cost
PHE-based [5]	$36 * N * \sigma$ ($\sigma = 1024$)	$18 * N$ operations over Z_{2^σ} ($\sigma = 1024$)
SMC-based [27]	$480 * (N - 1)$ (with 32-bit data)	3^N operations over $Z_{2^{32}}$
FHE-based [30]	0	N operations over Z_{2^σ} ($\sigma > 10^6$)
Our work	$(4N + 3) * \sigma$ ($\sigma = 1024$)	$(4N + 3)$ operations over Z_{2^σ} ($\sigma = 1024$)

Note: σ is the basic parameter and is different in each scheme; N : the number of provided data;

ACKNOWLEDGMENTS

This work is sponsored by the National Key Research and Development Program of China (grant 2016YFB0800704), the NSFC (grants 61672410 and U1536202), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), the Fundamental Research Funds for the Central Universities (grant JBG161509), the 111 project (grants B08038 and B16037), Academy of Finland (grant 308087), and the AXA Research Fund.

REFERENCES

- [1] A. Belle, R. Thiagarajan, S. Soroushmehr, F. Navidi, D. A. Beard, and K. Najarian, "Big data analytics in healthcare," *BioMed Res. Int.*, vol. 6, 2015, Art. no. 370194.
- [2] J. J. Stephen, S. Savvides, R. Seidel, and P. Eugster, "Practical confidentiality preserving big data analysis," in *Proc. 6th USENIX Workshop Hot Topics Cloud Comput.*, 2014, pp. 10–10.
- [3] B. Wang, M. Li, S. S. Chow, and H. Li, "A tale of two clouds: Computing on data encrypted under multiple keys," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2014, pp. 337–345.
- [4] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.
- [5] X. Liu, R. Choo, R. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Dependable Secure Comput.*, vol. PP, no. 99, 2016, doi: 10.1109/TDSC.2016.2536601.
- [6] X. Liu, R. Deng, W. Ding, R. Lu, and B. Qin, "Privacy-preserving outsourced calculation on floating point numbers," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 11, pp. 2513–2527, Nov. 2016.
- [7] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *IACR Cryptology ePrint Archive*, vol. 2014, 2014, Art. no. 331.
- [8] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *J. Netw. Computer Appl.*, vol. 42, pp. 120–134, 2014.
- [9] A. Khedr and G. Gulak, "SecureMed: Secure medical computation using GPU-accelerated homomorphic encryption scheme," *IEEE J. Biomed. Health Inf.*, Jan. 2017, doi: 10.1109/JBHI.2017.2657458.
- [10] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innovations Theoretical Comput. Sci. Conf.*, 2012, pp. 309–325.
- [11] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010.
- [12] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Adv. Cryptology-EUROCRYPT*, 2010, pp. 24–43.
- [13] V. C. Hu, T. Grance, D. F. Ferraiolo, and D. R. Kuhn, "An access control scheme for big data processing," in *Proc. Int. Conf. Collaborative Comput.: Netw. Appl. Worksharing*, 2014, pp. 1–7.
- [14] Z. Yan, W. Ding, V. Niemi, and A. V. Vasilakos, "Two schemes of privacy-preserving trust evaluation," *Future Generation Comput. Syst.*, vol. 62, pp. 175–189, 2015.

- [15] C. Huang, Z. Yan, N. Li, and M. Wang, "Secure pervasive social communications based on trust in a distributed way," *IEEE Access*, vol. 4, pp. 9225–9238, 2016.
- [16] Z. Yan, X. Li, M. Wang, and A. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 485–498, Jul.–Sep. 2015.
- [17] Z. Yan, X. Li, and R. Kantola, "Controlling cloud data access based on reputation," *Mobile Netw. Appl.*, vol. 20, no. 6, pp. 828–839, 2015.
- [18] W. Ding, Z. Yan, and R. H. Deng, "Encrypted data processing with homomorphic re-encryption," *Inf. Sci.*, vol. 409–410, pp. 35–55, 2017.
- [19] E. Ayday, J. L. Raisaro, J.-P. Hubaux, and J. Rougemont, "Protecting and evaluating genomic privacy in medical tests and personalized medicine," in *Proc. 12th ACM Workshop Privacy Electron. Soc.*, 2013, pp. 95–106.
- [20] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Trans. Sens. Netw.*, vol. 5, no. 3, 2009, Art. no. 20.
- [21] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 200–214.
- [22] Q. Li, G. Cao, and T. La Porta, "Efficient and privacy-aware data aggregation in mobile sensing," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 2, pp. 115–129, Mar. 2014.
- [23] Q. Li and G. Cao, "Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2013, pp. 60–81.
- [24] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2013, pp. 111–125.
- [25] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. 18th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2011, pp. 1–17.
- [26] D. Bogdanov, R. Talviste, and J. Willemson, "Deploying secure multi-party computation for financial data analysis," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2012, pp. 57–64.
- [27] L. Kamm and J. Willemson, "Secure floating point arithmetic and private satellite collision analysis," *Int. J. Inf. Secur.*, vol. 14, no. 6, pp. 531–548, 2015.
- [28] D. Bogdanov, "Sharemind: Programmable secure computations with practical applications," PhD dissertation, University of Tartu, Estonia, 2013.
- [29] J. H. Cheon, et al., "Batch fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2013, pp. 315–335.
- [30] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Exploring the feasibility of fully homomorphic encryption," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 698–706, Mar. 2015.
- [31] L. Morris. "Analysis of partially and fully homomorphic encryption," (2017). [Online]. Available: <http://www.liammorris.com/crypto2/Homomorphic%20Encryption%20Paper.pdf>
- [32] X. Liu, R. H. Deng, Y. Yang, H. N. Tran, and S. Zhong, "Hybrid privacy-preserving clinical decision support system in fog-cloud computing," *Future Generation Comput. Syst.*, vol. 78, pp. 825–837, 2018.
- [33] Z. Yan, W. Ding, and H. Zhu, "A scheme to manage encrypted data storage with deduplication in cloud," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2015, pp. 547–561.
- [34] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2008, pp. 127–143.
- [35] W. C. Garrison III, A. Shull, S. Myers, and A. J. Lee, "On the practicality of cryptographically enforcing dynamic access control policies in the cloud," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 819–838, doi: [10.1109/SP.2016.54](https://doi.org/10.1109/SP.2016.54).
- [36] T. Zhu, W. Liu, and J. Song, "An efficient role based access control system for cloud computing," in *Proc. IEEE 11th Int. Conf. Comput. Inf. Technol.*, 2011, pp. 97–102.
- [37] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.
- [38] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
- [39] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.
- [40] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.
- [41] Z. Wan, J. E. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 2, pp. 743–754, Apr. 2012.
- [42] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Adv. Cryptology—EUROCRYPT*, 1999, pp. 223–238.
- [43] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Proc. Adv. Cryptology—ASIACRYPT*, 2003, pp. 37–54.



Wenxiu Ding received the BEng degree in information security from Xidian University, Xi'an, China, in 2012. Now, she is working toward the PhD degree in information security in the School of Telecommunications Engineering, Xidian University. She was the research assistant in the School of Information Systems, Singapore Management University from 2015 to 2016. Her research interests include RFID authentication, privacy preservation, data mining and trust management.



Zheng Yan (M'06, SM'14) received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from Xi'an Jiaotong University, Xi'an, China, in 1994 and 1997, respectively, the second MEng degree in information security from the National University of Singapore, Singapore, in 2000, and the licentiate of science and the doctor of science in technology in electrical engineering from the Helsinki University of Technology (current Aalto University), Helsinki, Finland, in 2005 and 2007.

She is currently a full professor with the Xidian University, Xi'an, China and a visiting professor and an academy research fellow with the Aalto University, Espoo, Finland. Her research interests include trust, security and privacy, as well as data mining. She is serving as an associate editor of the *IEEE Internet of Things Journal*, the *IEEE Access*, the *Information Sciences*, the *Information Fusion*, the *Journal of Network and Computer Applications*, the *Soft Computing*, the *Security and Communication Networks*, etc. twelve journals. She serves as an organization and program committee member for numerous international conferences. She is a senior member of the IEEE.



Robert H. Deng is AXA chair professor of Cybersecurity and director of the Secure Mobile Centre, School of Information Systems, Singapore Management University. His research interests include the areas of data security and privacy, cloud security and Internet of Things security. He received the Outstanding University Researcher Award from National University of Singapore, Lee Kuan Yew fellowship for Research Excellence from SMU, and Asia-Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. His professional contributions include an extensive list of positions in several industry and public services advisory boards, editorial boards and conference committees. These include the editorial boards of the *IEEE Security & Privacy Magazine*, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Information Forensics and Security*, the *Journal of Computer Science and Technology*, and Steering Committee chair of the *ACM Asia Conference on Computer and Communications Security*.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Secure Dependency Enforcement in Package Management Systems

Luigi Catuogno¹, Clemente Galdi¹, and Giuseppe Persiano¹

Abstract—Package management systems play an essential role in pursuing systems dependability by ensuring that software is correctly installed and kept up-to-date according to vendor-defined installation policies. Circumventing such policies could make the system unhealthy and insecure and can constitute a serious security threat. In many application scenarios, e.g., distribution of commercial software, the confidentiality of the software must be guaranteed against non-authorized players. In some cases, the installation policy itself is considered a sensitive information, e.g., when it reveals required hardware in military contexts. In this paper we address the problem of strongly enforcing software dependencies in package management systems, to prevent that a malicious user forces the system to install any package despite its requirements are not completely fulfilled. The enforcement is *strong* in the sense that the encrypted software package cannot be even decrypted if the dependencies are not satisfied. Once a new package is decrypted and installed, our protocol *non-interactively* updates the key material on the target device. This key update will allow the decryption of further packages that depend on the newly installed one. We further present “policy-hiding” variants of our protocol. Finally we provide an experimental evaluation of the system performance.

Index Terms—Package management systems, secure software update, dependency enforcement

1 INTRODUCTION

SOFTWARE deployment is a critical activity within the life cycle of computer application systems. The tasks of distribution, installation, and update of software components require to be carefully designed and managed in order to keep the whole system safe throughout their execution. A fault occurring during any of the above tasks might prejudice the system integrity and trustworthiness.

Distribution facilities are required to not disseminate malformed or corrupted components as well as deployment agents are required to install/update a component on the system only if it fulfils the requirements posed by its vendor. For example, an application might come with the list of those required libraries that should be already present on the system, as well as the system could enforce any kind of software updates policy concerning of version, freshness, provenance and so on. Generally, such properties and requirements are referred to as *dependencies*.

Dependencies link different system components so that it is possible to identify which are the components that are affected by the installation or update operations. For this reason, ensuring integrity and coherence of dependencies through deployment tasks is a crucial activity which is

pursued at every step of the software lifecycle, as any failure during such a process may have disastrous consequences. Indeed, recent studies [2] identify dependencies breaking (during component deployment tasks) as a major cause of systems malfunctioning.

In many scenarios, the software itself might be considered sensitive information, e.g., in military applications. In these contexts, it is crucial to preserve software confidentiality w.r.t. users that are *not* allowed to install it. Furthermore, it might be the case that the confidentiality of the dependencies themselves should be protected as they may leak information about the software, e.g., the requirement for the driver of a *weapon* for the drone reveals the nature of the mission.

The purpose of the system we present is twofold.

Legitimate user security. Our system guarantees authenticity, integrity and freshness for legitimate users. This means that a given software is installed on an untampered device if and only if (a) It is authentic; (b) unmodified; and (c) all the required dependencies are properly satisfied.

Prevent unauthorized installations. A malicious user that is able to collect ‘files’ from different devices will not be able to install a given software even if the union of all collected files satisfies the required dependencies.

We consider software deployment within modern component based applications, including operating systems where deployment is mainly performed through either public or private networks. Deployable units, such as “packages”, “updates”, “patches”, are *produced* by a *deployment server*, *published* by a set of *mirror servers* and then *downloaded and installed/applied* by *client machines* either automatically or manually. This raises new issues related to security aspects. Indeed, adversaries could threaten their targets by exploiting any vulnerability throughout the whole software distribution

- L. Catuogno is with Dipartimento di Informatica, Università degli Studi di Salerno, Fisciano, Salerno 84084, Italy. E-mail: luicat@dia.unisa.it.
- C. Galdi is with Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Università degli Studi di Napoli “Federico II”, Napoli 80138, Italy. E-mail: clemente.galdi@unina.it.
- G. Persiano is with Dipartimento di Scienze Aziendali - Management & Innovation Systems, Università di Salerno, Fisciano, Salerno 84084, Italy. E-mail: giuper@dia.unisa.it.

Manuscript received 15 Mar. 2017; revised 7 Nov. 2017; accepted 22 Nov. 2017. Date of publication 27 Nov. 2017; date of current version 18 Mar. 2020. (Corresponding author: Clemente Galdi.)

Digital Object Identifier no. 10.1109/TDSC.2017.2777991

chain. In this way, a fraudulent deployment/mirror server, by impersonating a legitimate one, could distribute malformed/counterfeit packages. Similarly, by having the control of any network segment, an adversary could interfere with the deployment protocols, propagating misleading information and messages in order to delay or avoid the installation/update of certain components, so that the target systems remain in or pass to an unsafe or faulty state.

Exploiting software vulnerabilities or network ones have been widely addressed in the literature though, few solutions face both aspects as a whole. Indeed, in many cases, security features are designed as wrappers for existing deployment facilities, so that the formers encompass client-server mutual authentication and communication confidentiality, while the latter have in charge the verification of satisfiability and consistence of dependencies. However, as noticed above, both aspects influence each other as, on one hand, malicious protocol deviations may trigger unsafe deployment actions and on the other hand, wrong installation/update procedure may lead to the appearance of new security breaches.

In this paper we present a comprehensive mechanism for security and reliability in software deployment, focusing the attention on the scenario depicted by “Package Management Systems” as a special case of software deployment systems.

We consider the case in which *multiple independent* software vendors provide encrypted software *packages* along with their *installation policies*. Our system guarantees on one hand the confidentiality, integrity, freshness and authenticity properties of the packages. On the other hand, it allows a *strong enforcement* of dependencies in the sense that if some required dependency has not been installed using our system the package cannot even be decrypted and, consequently, cannot be installed. Each new package provides a number of *features*, each associated to a *random feature key* that are used to encrypt every future package that depends on the current. The same feature can be provided by multiple vendors. In this case, each vendor will provide its random feature key and we assume the existence of a naming strategy that allows the unique identification of features. In this respect, this protocol provides a *non-interactive* procedure that allows each device to update the set of feature keys associated to the installed packages.

We then present an extension of our protocol in which the installation policy can be hidden from the user. In other words, if the user is missing some of the prerequisite for the installation, she cannot decrypt the package and, at the same time, she is not able to say which package she is missing.

Our protocols constitute a sharp improvement with respect to previous solutions in which either package security and dependencies are *managed separately*, or target systems are essentially *static*, or software vendors have to cooperate to generate packages keys, i.e., software vendors *cannot be independent*, or the user and the vendor have to run an interactive protocol that is either *inefficient* or *leaks the device profile*, i.e., the list of installed packages.

2 RELATED WORKS

One of the first papers addressing security issues in software deployment via Internet is [3], where the author suggests to

introduce a trusted third party (*certifier*) who creates digital certificates for software vendors/developers (*issuers*). Each time a new package is released, the issuer provides the certifier with a signed certificate for the software that is then signed by the certifier. The user who wishes to install the package (*consumer*) is enabled to verify the authenticity of the software by verifying the certifier’s signature.

Such a solution is extremely simple and flexible and it is used as a baseline for modern package managers. Nevertheless, it has been shown in [4], [5], that it is potentially prone to a number of vulnerabilities due to the improper use of cryptographic primitives. In [6] the authors show that several major package management systems are vulnerable to *man-in-the-middle* attacks. In particular, [5] highlights the threat of malicious mirrors and third-party components in the distribution network (e.g., HTTP proxies).

In [7], authors emphasises the importance for a package management system to feature an effective content authentication mechanism. A secure software distribution infrastructure, leveraging run-time application integrity verification (*code signing*), is envisaged in [8]. Microsoft Windows systems introduced the Authenticode [9] technology to perform package authentication by means of code signing. Similarly, the Android OS features a code signing mechanism partially inherited from the Java framework [10]. However in [7] authors warn about the weak or non-existent resilience of some of such solutions to possible key disclosure and the lack of “trust revocation” mechanisms.

Authors in [11] address the issue of confidentiality of software updates where package distribution is carried out through partially or totally untrusted multiple mirrors. The paper introduces a scheme to manage package downloads without releasing information about the requested software. The scheme implements a Private Information Retrieval (PIR) [12] scheme. However, the solution presented in this paper has the strong requirement that each software download has to be executed by concurrent communication with multiple repositories.

Several works highlights that package dependencies remains a major matter of concern [13]. The RedHat’s Package Manager (RPM) [14], which can somehow be regarded as the archetype of the major current package management systems, has suffered from serious problems related to a weak mechanism to validate and enforce dependencies among different packages [15]. In particular, a relation between dependence inconsistency and security vulnerabilities has been put forth in [16].

In [17] the authors face the problem of achieving software confidentiality in a context in which the system provides updates to millions of devices through a cache-enabled network. In the envisioned scenario, client devices are partitioned in *classes* according to their own distinguishing set of *device attributes* (e.g., processor, OS, ...). The proposed mechanism to guarantee software confidentiality is built on top of a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme [18].

A preliminary version of this paper appeared as [1], where the key material is associated to each *package*. In this extension we associated keys to *features*. This choice allows a more flexible definition of dependencies.

2.1 Solutions and Issues Using Current ABE Schemes

Attribute Based Encryption (ABE) schemes [19] allow a user to decrypt a given ciphertext only if a set of attributes matches the ones described by the decryption policy.

ABE comes in two flavours. In Key-Policy ABE (KP-ABE) the decryption is possible if the attributes encoded in the cipher text satisfy the decryption policy specified by the secret key of the user. In Ciphertext-Policy-ABE, or CP-ABE, the decryption is possible if the attributes encoded by the secret key of the user match the decryption policy specified by the cipher text.

In this context, as it has been done in [17], a possible solution is to use CP-ABE scheme to encrypt the software package before its distribution. In this way the vendor can define the software decryption policy in the moment in which the package is published. At the same time there is no need to provide users with new key material. Indeed a user will be able to decrypt the package if she holds a set of attributes that meets the policy specified by the vendor.

This solution has a number of pros. First of all, starting from [20], there exist CP-ABE scheme in which the vendor can specify arbitrary (i.e., possibly non-monotonic) release policies. Notice that, in the context of software distribution, non-monotonic access structures can be used to describe *conflicts* between packages, i.e., the requirement that one package can be installed only if another one is *not installed*. Unfortunately, non-monotonic CP-ABE schemes require that the user receives a key element for every possible attribute. Since, in our interpretation, attributes correspond to software packages, this means that each user/device should receive a key element for *each possible package*.

Furthermore, the existence of Distributed CP-ABE schemes, first introduced in [21], allow software vendors to distributively compute the secret keys representing the attribute each user holds.

Thus, in a *static* setting in which the user attributes do not frequently change over time and in a setting in which either there exists a single software vendor or a few ones that are willing to cooperate, the usage of CP-ABE can be used to solve efficiently the problem of secure software distribution.

In this paper we consider a more *dynamic scenario*. Each time an administrator/user installs a new package on a device, such a software enriches the device with a new set of *attributes*. This is immediate, for example, in the case in which the installed package is a library. Indeed, the presence of the new library makes possible the installation of all the software packages that depend on it and whose installation was impossible before. In this respect, the installation of *each new software* modifies the set of attributes of a given system. If we look at the key management problem from this point of view, we can claim that it cannot be considered at all a static process.

It is thus crucial to provide efficient procedures for the update of the key material that users hold in response of a software installation. Unfortunately, in existing CP-ABE schemes either there must exist a single trusted centralised authority that generates the keys for each user or there are multiple, possibly independent, authorities [22] but each user has a unique *Global Identifier (GID)*. In the latter case, the user key generation requires the user to provide the GID

and all his attributes to the authority. This, in turn, would allow a complete profiling of the user device that is, in principle, a serious privacy issue and constitutes by itself a security threat. In PPDCP-ABE [23], the authors try to mitigate information leakage. Unfortunately, in this scheme the secret key generation algorithm requires interaction between the authority and the user.

3 PACKAGE MANAGEMENT

At glance, software deployment is the complex of activities pursuing components retrieval, installation and configuration on a computer system. Such components include software with many different purposes such as implementing system-wide services, features and tools (such as kernel modules, device drivers, dynamic libraries); new user applications rather than *updates* and *upgrades* for components already present on the system. Nowadays, major operating systems, as well as several frameworks and application suites, are conceived to operate within a *Package Management System (PMS)*, a distributed infrastructure which carries out the activities related to the deployment of software components. Main PMS activities are briefly described in the following paragraphs.

Once its development process is completed, a new version of a certain component is made available. The developer/vendor (*issuer*) releases a deployment unit (*package*) that consists of the component itself, along with the information to verify and install it (*metadata*). Packages can be published on the issuer's repositories as well as disseminated through a set of third-party *mirrors*, each of which replicates the content of the official repository.

Metadata have the twofold purpose of: (a) unambiguously identifying the component, its version, etc. and (b) declaring a set of properties and requirements (including dependencies) that have to be satisfied in order to correctly install the component. Metadata can be either shipped along with the software they refer to, or distributed separately. An official "components directory" is generally provided to publish at least metadata of the released package along with the link to the repository from which it can be downloaded.

The owner/administrator of the target system (*consumer*), is enabled to search for the requested component by querying the directory. Query fields include package's name, version, target system etc. Once she has obtained the information about the package, the consumer may choose whether or not to install the suggested components, according a *local* policy which takes into account several package's properties such as release date, freshness, provenance, license terms etc. Usually, the consumer is provided with a dedicated set of utilities that implement all deployment tasks. In the following such utilities are referred to as *package manager*.

The installation process concerns adding any new components to the target system that provide new features, whereas updates and upgrades that are about changing the setup of already installed components. Once a package is downloaded, the package manager verifies its integrity in order to identify any transmission error before the installation process begins. Furthermore, the authenticity of the package should be checked, in order to prevent the installation of counterfeit code.

Then, the package manager evaluates the package's *pre-installation* requirements, that mainly concern of target device properties (e.g., OS type/version, processor) and dependencies to already installed packages. Package management systems feature a local *package inventory* that stores the metadata of every installed package. The package manager uses it to check the satisfiability of dependencies. Eventually, metadata of the incoming package are added to the inventory. If all requirements are satisfied, the package manager copies all files contained in the package to their destination.

3.1 Security in Package Management Systems

Following [5], in order to improve their security, software distribution systems started to redefine the distribution protocols. In particular, the following guidelines are implemented by secure distribution services:

Package Authentication. Guarantee the (actual) usage of digital signatures.

Multiple protection levers. Provide, if possible, multiple levels of protection. Most repositories also provide a special *root metadata* file that stores the position in the file system and a secure hash for each package in the repository. The signature of root metadata coupled with the package signature or (package) metadata signature can provide different levels of security and usability.

Freshness Verification. Verify the *freshness* of metadata files. Outdated metadata files, even if such files are signed, might induce the user to install insecure software packages.

Multiple sources. Execute software updates from multiple sources, e.g., by checking the metadata on one repository and downloading the software package from another one. This procedure might allow the identification of malicious repositories.

4 SYSTEM MODEL AND REQUIREMENTS

We consider a model in which we distinguish three different players. Our model resembles the current Linux software distribution system. We will also describe the security requirements guaranteed by our system.

4.1 System Model

Software distributions/updates are carried out by means of a Distribution Server (DS) and possibly a set of Mirror Server (MS). The role of the Distribution Server is to generate properly formatted and encrypted software packages, containing the actual software to be installed, the installation policy, i.e., the specification of the set of required packages, along with the software metadata. Furthermore, the DS has to manage the timely and efficient distribution of the encrypted packages to the set of Mirrors. The Distribution Server is managed by the software vendor and it will be considered trusted. The role of the Mirror Servers is to answer requests sent by users and provide the required packages. The Mirror Servers are typically managed by third parties and, for this reason, MSes will be considered untrusted. Finally, the Users can query either one MS or the DS directly. As suggested in [5], and as it is currently done by some package distribution systems, the user might download for security reasons the metadata file from the DS while, for performance issues, the download of the software package is done via one MS.

The target device features a *legacy operating system* (LOS) and a *package management subsystem* that provides all functionalities related to the decryption and the evaluation of the installation policy and maintenance of the database of *feature keys*. The LOS features both normal and privileged users, whereas the *PMS administrator* is in charge of the PMS configuration and management. LOS and PMS are assumed to be mutually separated and communicate by means of a set of APIs. LOS and PMS administrators are two distinct roles. This should be not regarded as an unnatural assumption. For example, Windows Operating Systems, feature two privileged accounts: "administrator" and "SYSTEM". The former is intended to be used by the system manager for performing every operation except those affecting system's integrity, e.g., programs setup and removal. The latter can only be used non-interactively for package management and through a strictly manual procedure.

Package processing outputs *contents* (i.e., executables, configuration files, libraries etc. composing the installed software) and *metadata* (i.e., the information related to package version, author, release notes, verification tokens, etc.) along with some "private" metadata, concerning key material used to decrypt installation policies etc. (feature keys). Package contents are stored on the LOS filesystem, package metadata are stored in the PMS.

In facts, the PMS relies on a private secure storage where feature keys (along with any package-related private or verification information) are stored and never made available to any other system component (including LOS administrators).

Currently, plenty of technologies are available to implement such PMS feature. The secure storage can be built on top of cryptographic tamper-resistant devices such as SIM cards or other so-called Secure Elements at large, either embedded or plugged on-demand into the platform. On the other hand, target systems are required to ensure the trustworthiness of the PMS operation. Again, to this end, plenty of solutions are currently on the shelf, e.g., leveraging on Trusted Computing-powered operating systems or the forthcoming Trusted Execution Environment architecture where the PMS along with its key store naturally fits as "Trusted Application" [24].

4.2 Security Requirements

We define in this section the security requirements the proposed system guarantees.

Confidentiality. One of the key issues in software distribution is the confidentiality of the software. We will focus on the software protection from the moment in which the package is created by the issuer to the moment in which it is decrypted on the target device. Indeed, after its decryption, the software might be maliciously distributed at will.

Authenticity and Integrity. Another fundamental problem in software distribution/update is guaranteeing the authenticity and integrity of the installed software. The proposed solution will allow each user to *locally* verify that the software that she has downloaded meets such properties.

Freshness. As stated in the previous section, one of the key requirement for secure software update is guaranteeing the freshness of the packages. As an example consider the update of the database containing virus definitions. In this case, the freshness of the software component is crucial for

guaranteeing the system protection. Typically a new *fresh* package outdates all its previous versions/releases. We do not provide a formal definition of freshness as our system allows its formulation as an arbitrary combination software issuer and user defined policies. Finally, each issuer has the possibility/duty to decide its own policy for defining freshness for its own products.

Strong dependencies enforcement. We require that an adversary is not able to force the installation of a software on a system that does not meet all the prerequisites. The enforcement is *strong* in the sense that, if some required dependency has not been installed, the package cannot even be decrypted.

4.3 Adversary Model

In this paper we consider the case in which the Distribution Server is trusted by the users. Notice that, although we consider the case in which multiple independent distribution servers exist, from the point of view of the adversary, the existence of one or multiple of such servers is immaterial. Indeed each DS is assumed to work independently from the others. The model also assumes the existence of Mirror servers whose only role is to store encrypted packages generated by the DS and forward them to the users whenever they require it. The Mirror Servers are untrusted, as they might try to gain confidential information from the stored packages or send corrupted packages to the users. As in [17], the adversary has complete access to the communication network, i.e., she can manipulate, create or duplicate legitimate messages.

An adversary has LOS administrator privilege (including full access file systems and networking facilities). In particular the adversary can: (a) access and modify every file included in the package content (i.e., any file or component once it has been installed); (b) “manually” install (import) a package by copying its content as it is installed on a different device and (c) control/deviate the connection with the package repositories, in order to force the PMS to install malicious packages carrying fake features.

However, the adversary cannot alter the behaviour of the package management subsystem itself. That is, she can neither read, insert or delete feature keys nor modify vendors’ public keys nor modify any other information stored by the package management subsystem. To this aim, the system uses a tamper-proof secure storage area, which we refer to as the *package inventory* that is readable and modifiable only by the PMS.

The main goal of the adversary is to maintain/translate the target device in/to a state in which certain “malicious” activities are possible or certain features are illegitimately available. More precisely, the adversary aims at making the system either to install a package despite the system does not satisfy all its dependencies or to deny the installation of a package by “simulating” a dependency failure or forge the image of the device setup, in order to pursue any vendor support contract and license infringements.

4.4 A Note on the Application Scenario

As stated in the introduction, the purpose of our system is twofold. On one hand, it guarantees the authenticity and integrity of the package that is installed on a secure system.

On the other hand it forbids, even to the administrator, the installation of a package for which dependencies are not met. The latter requirement can be intended in one of the following ways: The package to be installed depends on a package that (a) has never been installed on the system. or (b) has been installed on the system but the adversary has modified it, *after its installation*, e.g., by deleting files. We completely cover case (a) as our system prevents the installation.

We point out that verifying the integrity of a package *after its installation* (e.g., at run-time) is not up to the PMS, as more effective solutions are available to this end [25]. In particular, we mention IMA [26], which is one of the earliest and most influential proposals in this field, along with its proposed improvements. IMA extends the *chain of trust* beyond the components involved in the bootstrap process, to the whole platform’s run-time.

Our PMS strengthens dependency enforcement and provides a mechanism to ensure package confidentiality in case its dependencies are not satisfied. Moreover the PMS, *before* installing a new package, verifies the integrity of those already installed features it depends upon. Having done so, its job is over. Nevertheless, our system does not require to be used necessarily in conjunction with any integrity verification tool, though it could give them an added value, as it could provide a further criterion to contain their verification scope.

In our system, features and dependencies form a hierarchy that roots in a set of features that do not explicitly depend on any other features. However every single feature implicitly depends at least on the base system, which includes the PMS itself. In our infrastructure, we made explicit this dependency in every package, so that, once the base system installation is completed, the PMS is already configured and its features keys repository contains at least the base system’s key (feature key zero). Therefore, packages which *apparently* have no requirements are actually encrypted with their package key which is, in turn, encrypted with such a key.

The feature key zero is stored on a secure element (e.g., a smart card), it is never released to the administrator and is provided only to the Operating System installer at setup time, through a secure protocol. To this end, we rely on the base assumption that the target platform provides secure boot facilities (e.g., UEFI’s secure boot [27] and Global Platform’s TEE secure boot [28]).

5 PRELIMINARIES AND NOTATIONS

Secret sharing schemes are one of the building blocks we use in our solution. Let P be a set of players and let $D \notin P$ be a special player called the dealer. An access structure \mathcal{A} over P is a monotone collection $\mathcal{A} \subseteq 2^P$. The basis of an access structure \mathcal{A} , denoted by $\delta(\mathcal{A})$, consists of the collection of its *minimal sets*, i.e., $\delta(\mathcal{A}) = \{A \in \mathcal{A} \mid \forall A' \subset A, A' \notin \mathcal{A}\}$. In our description, each software package corresponds to a *player* in the secret sharing terminology while an installation policy corresponds to an access structure.

It is well known that secret sharing schemes exist only for monotonic access structures. There exists a natural correspondence between the family of access structures and monotone boolean formulae [29]. We will denote by \mathcal{A}_ϕ the

access structure, defined over a set P of n players, representing the monotone boolean formula ϕ defined over a set of n variables. Specifically, a set $A \subseteq P$ is authorised in \mathcal{A}_ϕ and only if it corresponds to a truth assignment for ϕ . We note that, for every access structure \mathcal{A} , its basis $\delta(\mathcal{A})$ can be trivially mapped to a DNF formula by associating each authorised set in $\delta(\mathcal{A})$ to a clause in the formula ϕ .

Formally, a secret sharing scheme [30] consists of a pair of efficient algorithms, a distribution algorithm $\text{Distribute}(\cdot, \cdot)$ and a reconstruction algorithm $\text{Reconstruct}(\cdot, \cdot)$. The $\text{Distribute}(\mathcal{A}, s)$ algorithm takes as inputs an access structure \mathcal{A} over a set P and a secret s and outputs a set of shares $S = (s_1, \dots, s_n)$, one for each player in P . The reconstruction algorithm $\text{Reconstruct}(\mathcal{A}, \hat{S})$ takes as inputs the access structure \mathcal{A} and the set $\hat{S} \subset S$ of shares held by the players in $\hat{P} \subset P$. If $\hat{P} \in \mathcal{A}$ then the reconstruction algorithm outputs the secret s otherwise it outputs a special symbol \perp . Secret sharing schemes that work for every access structure exist, e.g., [31], but they may result in inefficient distributions. At the same time efficient secret sharing schemes have been developed for specific classes of access structure, e.g., [32], [33], [34].

We assume that each software issuer has a key pair (Sk_V, Pk_V) that are used to generate digital signatures. Given a message m and a signature $\sigma = \text{Sign}_{Sk_V}(m)$, we denote by $\text{Verify}_{Pk_V}(m, \sigma)$ the verification algorithm that either accepts the signature or outputs a special symbol \perp . The protocol uses a symmetric encryption scheme. We will denote by $\text{Encrypt}_k(\cdot)$, (resp., $\text{Decrypt}_k(\cdot)$) the encryption (resp., decryption) function under the secret key k . We assume that the encryption scheme used in the protocol meets the following properties: $\forall m$ and $\forall k, \text{Decrypt}_k(\text{Encrypt}_k(m)) = m$ and, $\forall m, \forall k, k', k \neq k'$ it holds that $\text{Decrypt}_k(\text{Encrypt}_{k'}(m)) = \perp$. The functions $\text{CreateMetaData}()$ and $\text{ExtractMetaData}()$ are used to create and extract the metadata information for a given software bundle, respectively.

6 THE PROTOCOL

In this section we describe the proposed protocols. Our system consists of two procedures. The first one, executed by the Distribution Server, is used by the software vendor to create an encrypted and signed package E and its associated metadata M_E . Such package will be then distributed to the Mirror Servers. A second procedure, executed by the user, takes as input the signed package E and its metadata and either installs the software therein contained or rejects it.

6.1 The Protocol: An Informal Description

In our proposal, the set of ciphertexts that has to be decrypted consists of the set of encrypted software packages. A system should be able to decrypt a given package only if it holds all the necessary *attributes* that are needed to execute the software therein contained. Furthermore, the package itself should contain some information that allows to update the set of attributes. Such a modification should be done *non-interactively*.

Whenever a new software package is installed on a given system, we can consider the new capability provided by the software as a new *attribute* of the system. Following the idea underlying ABE schemes, this new attribute should be used to update the set of keys of the system in order to extend the set of ciphertexts that can be decrypted by the system.

In our solution each package p is identified by a unique name n . The package p is assumed to *require* a set of features $\mathcal{R} = \{f_1, \dots, f_m\}$ and to *provide* a set of features, $\mathcal{P} = \{\hat{f}_1, \dots, \hat{f}_m\}$. Each feature is associated to a random key which we call the *feature key*. The set of keys provided by the packages installed on the current system are stored in a *package inventory*, which we denote by \mathcal{I} . As stated in Section 4.3, the information in \mathcal{I} cannot be read or altered by the adversary. Every package p that depends on the feature f_i should be encrypted with a random *encryption key* r that can be reconstructed if and only if the feature key k_i , associated to f_i , is known.

Informally, the distribution protocol works as follows. Consider a package p that requires *all* the features f_1, \dots, f_m . In this case the installation policy can be described by an access structure called *m-out-of-m* threshold access structure and denoted by $\mathcal{T}(m, m)$. Let us denote by k_i the feature key associated to feature f_i . The vendor generates a random encryption key $r = r_1 + \dots + r_m$ and computes $e_i = \text{Encrypt}_{k_i}(r_i)$. In other words she encrypts the random subkey r_i of the *encryption key* r for package p with the *feature key* k_i of feature f_i . The vendor builds a software package p by putting together the software, all the necessary metadata and all features keys associated to the features $\hat{f}_1, \dots, \hat{f}_m$ provided by the package p . This package is encrypted with the encryption key r , i.e., $E_r(p) = \text{Encrypt}_r(p)$. The vendor publishes the encrypted package along with e_1, \dots, e_m , i.e., $E = ((f_1, e_1), \dots, (f_m, e_m), \mathcal{T}(m, m), E_r(p))$.

When the package p needs to be installed, the user downloads E . In this case, if the user holds all the feature keys k_1, \dots, k_m , associated to all the required features, she can (a) decrypt every e_i , (b) reconstruct the key r and (c) properly decrypt $E_r(p)$. If the system is missing any of the prerequisite, she will not be able to execute the decryption and, thus she will not be able to install the package. Once the package has been decrypted, the device holds the keys associated to the features provided by the installed package and, from this point on, it is possible to install the software packages that depend on them.

6.2 A Protocol for Monotone Formulae

The protocol described in the previous section solves the problem of enforcing prerequisite matching in software installation whenever the installation policy requires *all the features in the set* $\mathcal{F} = \{f_1, \dots, f_m\}$, i.e., the installation policy can be described by a single-clause DNF formula $\phi = (f_1 \wedge \dots \wedge f_m)$.

Clearly this type of formula is not expressive enough to describe arbitrary installation policies. As an example, it cannot express an installation policy that requires the presence of feature f_i in one to its allowed versions v_1, \dots, v_m , i.e., $f_{i,v_1} \vee \dots \vee f_{i,v_m}$. Notice that having the possibility of expressing this type of installation policy is crucial as this is a typical way to express backward compatibility w.r.t. *some* previous versions of a given functionality.

Let p be a software package and let ϕ its installation policy. In the following we will assume that ϕ is described by a *monotone* boolean formula. The formula can be seen as an access structure \mathcal{A}_ϕ defined over the set of required features $\mathcal{R} = \{f_1, \dots, f_m\}$. Let us denote by $\{k_1, \dots, k_m\}$ the feature keys associated to the required features in \mathcal{R} . Furthermore,

let $\mathcal{P} = \{\hat{f}_1, \dots, \hat{f}_{\hat{m}}\}$ be the set of features provided by the package p and let $\{\hat{k}_1, \dots, \hat{k}_{\hat{m}}\}$ the corresponding feature keys.

Notice that, from the practical point of view, if a feature $f \in \mathcal{R}$ is required for the installation of package p , it has to be the case that the vendor has already installed some package that provides the functionality f . This assumption is quite natural in contexts in which the software to be released has to be quality certified. For example, let us consider the Common Criteria Recognition Agreement [35], a *de-facto* standard for the certification of security-related products. CCRA software quality testing is executed on a specific software *configuration* or *set of configurations*. Roughly speaking, a configuration identifies, among others things, all software components, including the specific version of each external package the software uses, along with their implementation. We can thus safely assume that the software vendor knows the feature key k_f . Would it be otherwise, she could not have even created and tested her own software.

We further observe that for the functionalities $\hat{f}_i \in \mathcal{P}$, two cases may arise. Either a feature \hat{f}_i is a new functionality, implemented by p for the first time, or it already existed in some other package, e.g., in a previous version of the same package p . In the former case, the corresponding feature key \hat{k}_i is randomly generated at the package creation time. In the latter case, as for the keys associated to the required features, we can assume that they are known to the software vendor. In any case, we can assume that the software vendor is able to obtain all the feature keys associated to the features in \mathcal{R} and \mathcal{P} .

At package creation time, the set of keys associated to the features provided by the current software has to be embedded in the encrypted package. The software issuer extracts the feature keys associated to existing features from the package inventory. At the same time, she generates random feature keys for new ones and updates the inventory accordingly. Algorithm 1 describes the construction of such set of keys.

Algorithm 1. Generation of the Set of Provided Features Keys

function GENERATEPROVIDEDKEYS (\mathcal{I}, \mathcal{P})

$\mathcal{K} \leftarrow \emptyset$

$\hat{m} \leftarrow |\mathcal{P}|$

for $j \leftarrow 1$ to \hat{m} **do**

$\hat{k}_j \leftarrow \text{Extract_Key}(\mathcal{I}, \hat{f}_j)$

if $\hat{k}_j = \perp$ **then**

 Generate a random feature key \hat{k}_j

$\mathcal{I} \leftarrow \mathcal{I} \cup (\hat{f}_j, \hat{k}_j)$

end if

$\mathcal{K} \leftarrow \mathcal{K} \cup (\hat{f}_j, \hat{k}_j)$

end for

return $(\mathcal{K}, \mathcal{I})$

end function

The package creation process proceeds as follows. The software vendor generates a random encryption key r and shares it among the required features in \mathcal{R} by using an efficient secret sharing scheme for \mathcal{A}_ϕ . In this way, each feature $f_i \in \mathcal{R}$ will have an associated share s_i . Each such share is then encrypted with the feature key k_i , i.e., $e_i = \text{Encrypt}_{k_i}(s_i)$. The sequence

$R = (f_1, e_1), \dots, (f_m, e_m)$ will be included in the package. Intuitively, each device holding the feature keys associated to all the features required by authorized set in the installation policy will be able to recover r , otherwise she will obtain no information about it.

The vendor then generates a timestamp t and creates a package containing the software s , the software name n , the set of provided features keys pairs $\{(\hat{f}_1, \hat{k}_1), \dots, (\hat{f}_{\hat{m}}, \hat{k}_{\hat{m}})\}$, the timestamp t , a freshness interval length Δ , the software metadata M_s . This package is encrypted by using the encryption key r . The encrypted package, along with the sequence R of encrypted shares and the installation policy \mathcal{A}_ϕ are packed together to obtain the E .

The vendor creates metadata information for E by including the package name n , the timestamp t , the software metadata M_s and other required information for E , e.g., size, hash value, creation date, extra information, etc. Notice that once the package is encrypted, the package name, the timestamp and metadata M_s are bound to the software but, at the same time, they are not accessible without decrypting it. Since such information is also needed in the clear in order to speedup integrity checks, we add them both the E and M_E . Finally, the vendor signs separately E and M_E and send to the Mirrors. Algorithm 2 describes the protocol executed by the vendor.

Algorithm 2. Distribution Server

procedure CREATEPACKAGE($Sk_V, s, M_s, \mathcal{I}, \mathcal{R}, \mathcal{P}, \mathcal{A}_\phi, \Delta$)

 Generate a random encryption key r

\triangleright Key Sharing

$(s_1 \dots, s_m) \leftarrow \text{Distribute}(r, \mathcal{A}_\phi)$

 Parse \mathcal{R} as $\{f_1, \dots, f_m\}$

for $j = 1$ to m **do**

\triangleright Shares Encryption

$k_j \leftarrow \text{Extract_Key}(\mathcal{I}, f_j)$

$e_j \leftarrow \text{Encrypt}_{k_j}(s_j)$

end for

$t \leftarrow \text{TimeStamp}()$

\triangleright Encrypted Package Generation

$(\mathcal{K}, \mathcal{I}) \leftarrow \text{GenerateProvidedKeys}(\mathcal{P}, \mathcal{I})$

$p \leftarrow (n, t, \Delta, M_s, \mathcal{K}, s)$

$E \leftarrow \langle \langle (f_1, e_1), \dots, (f_m, e_m) \rangle, \mathcal{A}_\phi, \text{Encrypt}_r(p) \rangle$

$M_E \leftarrow \text{CreateMetaData}(n, t, \Delta, M_s, E)$

$\sigma_M \leftarrow \text{Sign}_{Sk_V}(M_E)$

$\sigma \leftarrow \text{Sign}_{Sk_V}(E)$

 Send $(E, \sigma), (M_E, \sigma_M)$ to Mirror Servers

end procedure

The key reconstruction algorithm takes as inputs the bundle E and the package inventory \mathcal{I} . The user extracts from E the sequence $(f_1, e_1), \dots, (f_m, e_m)$ and the installation policy \mathcal{A}_ϕ . She decrypts all the shares she can by using the feature keys contained in the local key inventory \mathcal{I} and, if possible, she reconstructs the encryption key r .

The installation procedure described by Algorithm 3 is executed by the user. It takes as inputs the metadata (M_E, σ_M) , signed by the issuer and downloaded either from the Distribution Server or from one Mirror Server; the signed package (E, σ_E) ; the issuer public key and the local package inventory. As a preliminary step, the user checks the signatures on these packages by using the verification key of the software vendor. The user checks that the bundle E matches the metadata information contained in M_E . Furthermore, as anticipated in Section 4.2, depending on the

local and on the *issuer* policies, she checks the freshness of the package. If any of the above checks fails, the installation is rejected. The key reconstruction algorithm is used to recover the decryption key and the software package is decrypted. If the decryption is successful, the user extracts the software metadata M_s from M_E and verifies their correspondence. In this phase, the procedure also checks, e.g., by using the metadata stored by the PMS, if the implementations of the features required by the package have not been altered. If all tests pass, she installs the software and adds the features keys contained in E to the package inventory.

Algorithm 3. User Installation Procedure

```

procedure INSTALL( $(M_E, \sigma_M), (E, \sigma), Pk_V, \mathcal{I}$ )
  if (Verify( $E, \sigma$ ) =  $\perp$ )  $\vee$  (Verify( $M_E, \sigma_M$ ) =  $\perp$ ) then
    Reject  $\triangleright$  Authenticity Verification
  end if
  if ( $E$  does not Match  $M_E$ )  $\vee$  ( $M_E$  Not Fresh) then
    Reject  $\triangleright$  Integrity and Freshness Verification
  end if
  Parse  $E$  as  $\langle\langle(f_1, e_1), \dots, (f_m, e_m)\rangle\rangle, \mathcal{A}_\phi, S\rangle$ 
   $r \leftarrow$  ReconstructKey( $\langle\langle(f_1, e_1), \dots, (f_m, e_m)\rangle\rangle, \mathcal{A}_\phi, \mathcal{I}$ )
   $p \leftarrow$  Decrypt $_r(S)$   $\triangleright$  Package Decryption
   $M_s \leftarrow$  ExtractMetadata( $M_E$ )
  if ( $p \neq \perp$ )  $\wedge$  ( $p$  matches  $M_s$ ) then
    Parse  $p$  as  $(n, t, \Delta, M_s, \mathcal{K}, s)$ 
    Check required features integrity on disk
    Install  $s$   $\triangleright$  Software Installation
     $\mathcal{I}' \leftarrow \mathcal{I} \cup \mathcal{K}$   $\triangleright$  Package inventory update
  end if
end procedure

```

6.3 Security Analysis

In this section we provide a security analysis w.r.t. the security requirements presented in Section 4.2.

Confidentiality. The confidentiality of the software in the encrypted package is guaranteed by means of a symmetric encryption scheme. Since we assume that such a scheme is secure, the confidentiality of package relies on the impossibility of extracting the encryption key r from the encrypted shares $\langle\langle(f_1, e_1), \dots, (f_m, e_m)\rangle\rangle$. Here, again, the security of each share is guaranteed by the security of the symmetric encryption scheme used to encrypt it and by the security of the package inventory. Furthermore, for every set $A \subseteq N$ such that $A \notin \mathcal{A}_\phi$, since secret sharing scheme is perfect, the user can obtain no information on the decryption key given all the decrypted shares associated to the packages in A . This means that the user can obtain the encryption key if and only if she holds all the feature keys associated to required packages.

Authenticity and Integrity. Software integrity is guaranteed by the use of secure hash functions in the generation of the metadata associated to the software and for the one associated to the encrypted package E . Secure signature schemes are used to guarantee the authenticity of the information that the Distribution Server sends to the Mirror Servers and that these ones forward to the Users.

Freshness. The encrypted package E and its associated metadata M_E include a timestamp t , generated by the Distribution Server, and the length of the validity period Δ . From the point of view of the issuer the package has to be

intended to be fresh in the interval $[t, t + \Delta]$. On the other hand the freshness of a package can be influenced by the installation policy of the user. For example, a PMS administrator may use the following policy: *Install security packages immediately but install other packages only when the version is stable*. For this reason we do not bind the freshness to the issuer-generated freshness interval but we allow the user to override this interval whenever its local policy requires to.

Strong dependencies enforcement. The user installation procedure guarantees the dependency enforcement under the assumption that the local package inventory has not been modified. Indeed, under such assumption, the user will not be able to decrypt the downloaded package unless she meets the issuer-defined installation policy. Furthermore, the procedure also verifies that feature implementations have *not* been altered *after* their proper installation. We stress that if we start from a safe system, the strong dependency enforcement guarantees that the installation does not bring the device in an unsafe state that, in turn, guarantees the inventory integrity. Furthermore, we can relax this assumption by requiring the administrator to sign the local inventory after each installation.

7 TWO POLICY-HIDING PROTOCOL EXTENSIONS

The protocols presented in the previous section assume that the installation policy is public. However, there might be cases in which the installation policy, or even part of it, should not be public. Consider, for example, the case in which the software package has to be installed on a flock of military drones. The installation policy might express, among other things, requirements regarding the set of devices that are required to complete the mission. In this case, the installation policy itself might release sensitive information, e.g., it is clear that the goal of a mission requiring a *weapon* device is different from the one requiring a *camera* device. In this context, our extensions hide (part of) the installation policy to every user who *does not* have the right to install the package.

In order to meet this additional security requirement, in this section we present two policy-hiding protocol extensions. The two extensions aim at hiding different information regarding the installation policy. More precisely, in the first extension the structure of the installation policy is public. On the other hand, either the adversary holds the feature key needed to decrypt a given share s_i generated during the package creation process or she cannot tell to which feature f_i that share is associated to. In the second extension, the formula structure itself will be hidden from the adversary.

7.1 Policy Hiding: An Overview

In order to provide policy-hiding protocols, we need to take special care of two different issues. First of all, the protocols should not release feature names in the clear in the package. Second, and more important, is the fact that secret sharing schemes assume the access structure to be public. Secret sharing schemes exist for *general*, i.e., arbitrary monotone, access structures and for *specific*, e.g., threshold, or graph-based, access structures. In the former case the reconstruction algorithm requires as input the access structure to properly combine the shares. General schemes sacrifice

efficiency (in terms of share size) for generality. In the latter case, the reconstruction algorithm might not require the access structure as input. On the other hand, since schemes are designed for specific classes of access structures, they are more efficient than general ones at the cost of losing generality and releasing the access structure topology.

In both cases, the protection of the installation policy is sacrificed in favour of efficiency considerations. Our idea underlying the protection of the policy is to use a less efficient solution in favour of a more secure one.

7.2 An Efficient Protocol for Partially-Hidden Policies

A first step toward policy protection is the *anonymization* of feature names. The package creation procedure includes feature names in the installation policy specification. The key idea is to remove feature names from the package. Notice that, such a protocol modification does not hide the installation policy but clearly hides the correspondence between an encrypted share and its required feature both to a legitimate user and to an adversary. Since references to feature names are missing, the user cannot recover the proper key by looking up in package inventory but she needs to scan such database and try to decrypt the current share with each key therein contained. At first sight, this might turn out to be quite inefficient as the decryption of each share takes a time that grows with the number of feature keys stored in the inventory. However, as we will see in the next section, in practice the number of features included in a typical system configuration rarely exceeds one thousand, so that the scanning time remain quite reasonable (about few seconds). Such decryption strategy works if the user is able to distinguish between the case in which she is using the correct key from the case in which the key she is using for the decryption is wrong.

Clearly, an adversary that receives a package, can recognise the structure of the installation policy and she can decrypt each share for which she knows the corresponding key. On the other hand, for each share for which she does *not* know the corresponding feature key, the adversary cannot say which is the required feature she is missing.

7.3 A Protocol for Fully-Hidden Policies

In order to further reduce the amount of information leaked by package, we present a protocol that also hides the structure of the installation policy. More specifically, we will use a *standardised* access structure defined over a set of *anonymized* feature names. Intuitively, standardised access structures cannot be used for secret sharing scheme designed for specific classes of access structures. At the same time, as before, anonymised feature names do require a linear search in the package inventory for retrieving the proper feature key, in place of the direct access given the feature name.

Let ϕ be the monotone formula defined over the variables associated to the required features $\mathcal{R} = \{f_1, \dots, f_m\}$. We first anonymise the set of names, by substituting each name in \mathcal{R} with its corresponding index i , i.e., $\hat{\mathcal{R}} = \{1, \dots, m\}$. We stress that this mapping is performed locally (and arbitrarily) by the dealer/software vendor. We then convert the formula ϕ into its Disjunctive Normal Form. In this way we transform a monotone boolean formula ϕ defined over \mathcal{R} into a

monotone boolean formula in DNF $\hat{\phi}$ defined over $\hat{\mathcal{R}}$. Given a DNF formula $\hat{\phi} = C_1 \vee \dots \vee C_k$, where $C_i = x_{j_1} \wedge \dots \wedge x_{j_l}$, it is possible [31], [32] to additively share the encryption key r , independently for each C_i , and obtain shares $s_{i,j_1}, \dots, s_{i,j_l}$. At this point, as in the previous protocol, each share s_{i,j_h} associated to feature j_h is encrypted with the feature key k_{j_h} , i.e., $E_i = (\text{Encrypt}_{k_{j_1}}(s_{i,j_1}), \dots, \text{Encrypt}_{k_{j_l}}(s_{i,j_l}))$. The Distribution Server publishes $E = ((E_1, \dots, E_k), \text{Encrypt}_r(p))$. Algorithm 4 describes the whole protocol executed by the Distribution Server. Notice that E does *not* contain a description of the installation policy in clear. The information that can be inferred by this representation is the number of clauses in the installation policy and the number of literals in each clause.

Algorithm 4. Anonimized Distribution Server

procedure ANONPKGCREATION($s, M_s, \mathcal{I}, \mathcal{R}, \mathcal{P}, \Delta, \mathcal{A}_\phi$)

 Generate a random encryption key r

 Parse \mathcal{R} as $\{f_1, \dots, f_m\}$

for all $A_i = \{j_1, \dots, j_{l(i)}\} \in \delta(\mathcal{A}_\phi)$ **do**

 Randomly select $(s_{i,j_1}, \dots, s_{i,j_{l(i)-1}})$

$s_{i,j_{l(i)}} \leftarrow r \oplus \bigoplus_{w=1}^{l(i)-1} s_{i,j_w}$

for $w \leftarrow 1$ to $l(i)$ **do**

$k_{j_w} \leftarrow \text{Extract_Key}(\mathcal{I}, f_{j_w})$

$e_{j_w} \leftarrow \text{Encrypt}_{k_{j_w}}(s_{i,j_w})$

end for

$C_i \leftarrow (e_{j_1}, \dots, e_{j_{l(i)}})$

end for

\triangleright Encrypted Package Generation

$t \leftarrow \text{TimeStamp}()$

$(\mathcal{K}, \mathcal{I}) \leftarrow \text{GenerateProvidedKeys}(\mathcal{P}, \mathcal{I})$

$p = (n, t, \Delta, M_s, \mathcal{K}, s)$

$E \leftarrow \langle (C_1, \dots, C_{|\mathcal{A}|}), \text{Encrypt}_r(p) \rangle$

$M_E \leftarrow \text{CreateMetaData}(n, t, \Delta, M_s, E)$

$\sigma_M \leftarrow \text{Sign}_{Sk_V}(M_E)$

$\sigma \leftarrow \text{Sign}_{Sk_V}(E)$

 Send $(E, \sigma), (M_E, \sigma_M)$ to Mirror Servers

end procedure

The above transformation forces the modification of the Key Reconstruction Phase in the installation procedure, too. Indeed, when the user downloads the software package, because of the anonymization of feature names, she does not know which installed key should be used to decrypt a given share. Thus the procedure has to decrypt each share with each installed key. Clearly since shares are random, we need to make again the loose assumption that the user can distinguish the decryption executed with the wrong key from a decryption executed with the correct key. Upon retrieving a software package, for each encrypted clause E_i the user tries to decrypt each encrypted share $\text{Encrypt}_{k_{j_1}}(s_{i,j_1})$ using all the keys she has in her local package inventory. If she fails in decrypting one share, she moves to the next clause. If she manages to decrypt all the shares in a given clause, she can reconstruct the encryption key for the package and install the software.

7.4 Security Discussion

The formula transformation *partially* solves the issues related to the publicity of the access structure since every access structure is now defined by a DNF formula. On the other hand, an adversary obtains information about the

number of clauses in the formula and their size. Furthermore, an adversary can infer information about the software packages she has locally installed, e.g., if the key k_a correctly decrypts a literal in a clause then the adversary knows that current package depends on feature f_a . We notice that this transformation tends to gain privacy by loosing efficiency. Indeed the transformation from an access structure to its corresponding basis might bring an exponential blow-up in the policy size, e.g., in the case of threshold access structure. Furthermore, the anonymization of package names forces the user to linearly search correct keys instead of directly accessing them.

Algorithm 5. Decryption Key Reconstruction Phase

```

function ANONYMIZEDRECONSTRUCTION( $(E, \sigma), \mathcal{I}$ )
  Parse  $E$  as  $\langle\langle C_1, \dots, C_{|\mathcal{A}|} \rangle\rangle, \text{Encrypt}_r(p)$ 
  for  $i \leftarrow 1, \dots, |\mathcal{A}|$  do ▷ Scan each clause
    Parse  $C_i = \langle e_1, \dots, e_{l(i)} \rangle$ 
    for  $q = 1, \dots, l(i)$  do ▷ Search decryption key
      for all  $k_w \in \mathcal{I}$  do
         $s_q \leftarrow \text{Decrypt}_{k_w}(e_q)$ 
        if  $s_q \neq \perp$  then
          break
        end if
      end for
    end for
  if (Correctly decrypted each share in  $C_i$ ) then
     $r \leftarrow \bigoplus_{q=1}^{l(i)} s_q$ 
    return  $r$ 
  end if
end for
return  $\perp$ 
end function

```

8 PERFORMANCE EVALUATION

In this section we present the experimental evaluation we have carried out for testing the performance of the proposed software installation system. Our system has an impact on two operations in the package management lifecycle, namely, package creation and installation. Both operations consists of two separate phases.

Key processing. During the first phase the system either shares the encryption key according to the installation policy or reconstructs it, given the encrypted shares, the installation policy and the feature keys. The performance of this phase heavily depends on the performance of our scheme. Indeed, the amount of additional information to be stored and the amount of time needed to run this phase depend on the protocol type and on the access structure to be implemented.

Package processing. During this phase the package is either created and encrypted or decrypted and installed. The (d)ecryption operation only depends on the size of the package. We notice that the package contains a specification of the installation policy, whose size depends on the specific protocol distribution protocol we use. On the other hand, as we will see, the size of this specification is negligible w.r.t. the size of the software contained in the package. For this reason, the time needed to execute this phase is *independent* from our protocols and we do not consider this phase in our evaluation.

8.1 The Experimental Setup

For the sake of an experimental evaluation, we tested the proposed software update system by extending the RPM Package Management System, one of the most deployed PMSes. Our experimental test set consists of all the packages contained in the following Linux Distributions: CentOS, Red Hat Enterprise Linux (RHEL), OpenSuse and Fedora. We have considered the installation policy in each package and classified it according to the measures we discuss below. We have considered two extreme cases. The former case corresponds to the point of view of a software vendor that needs to encode *all the packages in a given distribution*, by making each package backward compatible with some of its previous versions. The latter one, corresponds to the case in which the vendor needs to prepare a *core* distribution, i.e., a minimal set of packages that guarantee the system to operate properly.

The experiments have been carried out on a dedicated desktop PC powered by an Intel Core i5 CPU M460 at 2.53 GHz x4, equipped with 5GB Ram and a 1TB HD and running a Linux Ubuntu 15.10 32-bit. All policy-size experiments were conducted by ad-hoc Python 2.7 scripts leveraging python rpmlib version 4.12, while time-measurements were conducted by a Java application.

8.1.1 RPM File Format

Each RPM package is encoded in a single file, formatted according to the rpm format. Since its very first versions, RPM was designed to allow each package to (a) describe the set capabilities it requires to be executed, (b) list the set of capabilities it provides and (c) store and retrieve such capabilities from the RPM database.

The process implemented by the RPM package manager to build an rpm file first analyses the list of files to be packaged. It constructs the list of capabilities provided by the package and, most importantly, the list of capabilities required by the package. The required capabilities that correspond to dependencies are classified by RPM into *automatic* and *manual* ones. Automatic dependencies correspond to the set of shared libraries that are required by the current package and are computed by the package manager using the `ldd` command. Since *all* automatic dependencies are required to properly run the current package, their associated installation policy corresponds to a single-clause DNF formula.

Manual dependencies correspond to specific capabilities that the package creator might require. Examples of such capabilities are the requirements for a specific package or for a specific set of versions of a given package. Versions can also be specified by using the operators $<$, \leq , $=$, $>$ or \geq , e.g., `requires: foo >= 1.5`.

RPM format allows the specification of features that conflict with the current one, i.e., package that cannot be installed simultaneously with the current package. Conflicts correspond to the logical complement of required features and are represented by non-monotone formulas. The management of non-monotone formulas is currently under investigations.

8.1.2 RPM Installation Policies and Boolean Formulae

If we consider the boolean formulae that can be constructed by the RPM package manager, we can state that they only consist of conjunctions of the following types of formulas:

TABLE 1
Experimental Set Up

Distribution Name	Experiment Acronym	Full Distributions			Core Distributions				
		Ver.	#Packages	#Provides	#Requires	Ver.	#Packages	#Provides	#Requires
CentOS (Minimal)	CentOS-M	6-7	574	3,435	8,044	7	357	410	655
CentOS (Complete)	CentOS-C	4-5-6-7	20,737	122,312	261,896	7	169	673	823
Fedora	Fedora	20-21-22-23	11,298	200,467	142,640	23	145	264	541
OpenSuse	OpenSuse	11-12-13-13.2	19,444	175,181	229,503	13.2	1,267	6,558	9,002
RedHat Enterprise	RHEL	4-5-6-7	13,292	93,019	196,994	7	166	678	837

type-a. Automatic dependencies correspond to a formula consisting of exactly one DNF-clause that contains one literal for each required feature f_i ;

type-b. For each additional vendor requirement, a formula consisting of the disjunction of literals associated to the features $f'_{i,v_{i,1}}, \dots, f'_{i,v_{i,k_i}}$ corresponding to the different versions of the required feature f'_i .

In general, let us assume the (only) type-a clause consists of k literals. Furthermore, let c be the number of type-b clauses in the installation policy and let us denote by k_i , $i = 1, 2, \dots, c$ the number of literals in the i th clause. The installation policy generated by the RPM manager can be described by the following boolean formula

$$\hat{\phi} = \bigwedge_{i=1}^k f_i \wedge \bigwedge_{i=1}^c \left(\bigvee_{j=1}^{k_i} f'_{i,v_{i,j}} \right). \quad (1)$$

This formula can be used as is in the protocols described by Algorithms 2 and 3, i.e., when the installation policy is public or when we do require the partial hiding of the policy as described in Section 7.2. Notice that it is possible to store this formula by using space $|\hat{\phi}| = O(k + k_1 + \dots + k_c)$.

In case of fully-hiding protocols, described by Algorithms 4 and 5, we need to convert the installation policy in DNF, that corresponds to the following formula $\phi = \bigvee_{j_1=1}^{k_1} \dots \bigvee_{j_c=1}^{k_c} (\bigwedge_{i=1}^k f_i \wedge f'_{1,v_{1,j_1}} \wedge \dots \wedge f'_{c,v_{c,j_c}})$

The formula consists of $\prod_{i=1}^c k_i$ clauses, each with exactly $k + c$ literals. Thus the size of the formula can be computed as $|\phi| = O((k + c) \prod_{i=1}^c k_i)$.

8.1.3 Estimating the Policy Size

As stated before, $|\hat{\phi}|$ and $|\phi|$ express the number of literals in the formula associated to the installation policy. Specifically, the former corresponds to the public or partially-hidden policy, while the latter to the fully-hidden policy. For each literal we need to store a constant number of bytes, namely the pair (feature name, encrypted share). Thus the total size of the formula is linear in the number of literals.

In our experiments we have evaluated the above formulas in real-life operational settings for the test distributions CentOS, Red Hat Enterprise Linux, OpenSuse and Fedora. More precisely, for each test distribution we have evaluated the values of k , c and k_1, \dots, k_c by considering either 2 or 4 consecutive versions of each test distribution, as listed in Table 1.

To do so, we have first constructed a database containing the list of all pairs (Feature, Feature Version), that are provided by all the packages in all the versions of a given distributions in the test set. Given such a database, for each (distinct) package, it is possible to obtain the value of k as the number of required features for which a single version,

compatible with the installation policy, exists in the database. Similarly, the value of c is obtained by counting the number of required features for which multiple versions, compatible with the installation policy, exist in the database. Finally, the values k_1, \dots, k_c is computed by counting the number of different versions of each required features that is present in the database.

Let us consider, for example, the Fedora Server distribution. We have constructed the feature-version database by listing all features contained in the Fedora distributions 20, 21, 22 and 23. At this point, for each (distinct) package in any of the above distributions, we have computed the value c by counting the number manual requirements listed in the installation policy. Furthermore, for each such requirement, we have computed k_i by counting the number of features in the database that satisfy the requirement.

For example, assume that featureA version 3.2.9 is contained in Fedora Server 20 and 21. The same feature is contained in version 3.3.0 in Fedora Server 22 and versions 3.3.1 in Fedora Server 23. Furthermore, let us assume that featureA is required in its version $\leq 3.3.1$ in some package in Fedora Server 23. Thus the list of all pairs in the database that are compatible with this requirement is (featureA, 3.2.9), (featureA, 3.3.0) and (featureA, 3.3.1), that has cardinality $k_i = 3$.

8.1.4 Formula Size in the Protocol with Public or Partially-Hidden Policy

In order to guarantee the correctness of the protocol defined by Algorithm 3, the package needs to store (a) a specification of the formula defined by Equation (1) and (b) the encrypted shares for pairs (feature, version). In this case the formula can be succinctly represented by using $k + k_1 + \dots + k_c$ pairs (feature, version). Furthermore, the package will contain $k + k_1 + \dots + k_c$ encrypted shares, again, one for each pairs (feature, version) contained in the installation policy,

Table 2 contain the measurements over the different distributions in test set. Specifically, Table 2 contains the maximum, average number and the percentage of packages with a given range of required for the parameters c , k , i.e., the number of required features for which one or multiple versions exist in the feature database. Furthermore it contains the percentage of the number of clauses in the formula whose size lays in the given range. From the data reported in such table, we can notice that, for the vast majority of the packages in all the distributions' set, the size of the formula that corresponds to the installation policy is small, i.e., below 50 literals. In the worst case, the number of literals is below 531, that corresponds to a space of few kilobytes on disk.

Let us consider the time needed to execute the key sharing/recovery phase. In the case the policy is public, the

TABLE 2
Cumulative Percentage for the Expansion Parameters in the Public or Partially-Hidden Policies for Full and Core Distributions

Exp. Acronym	Full Distributions										Core Distributions		
	c			k			$ \hat{\phi} = k + k_1 + \dots + k_c$				$ \hat{\phi} = k + k_1 + \dots + k_c$		
	Av.	Max	0-10	Av.	Max	0-100	Av.	Max	0-50	0-100	Av.	Max	0-50
CentOS-M	0,90	8	100,00%	13,10	80	100,00%	15,14	87	97,91%	100,00%	16,60	62	98,04%
CentOS-C	1,07	32	99,75%	11,56	232	99,67%	17,28	531	94,12%	99,09%	12,37	62	98,82%
Fedora	0,85	38	99,81%	11,78	319	99,46%	16,46	437	95,18%	99,16%	17,88	50	100,00%
OpenSuse	0,94	44	99,13%	10,86	223	99,76%	15,64	252	95,04%	99,41%	15,17	74	98,18%
RHEL	1,06	32	99,71%	13,71	232	99,59%	19,75	522	92,70%	98,84%	12,63	62	98,80%

vendor/user needs to execute a number of (symmetric) share encryptions (resp., decryptions) that is upper bounded by the number of literals in the formula. Since such a number is upper bounded by few hundreds, such key sharing/recovery phase takes few milliseconds.

The situation becomes slightly more complex when the policy is required to be partially hidden. In this case, the encrypted share associated to each literal in the formula has to be decrypted by using each key in the local database. As previously highlighted, in practical applications, the number of installed features remains well below the number of all available features. Analogously, the amount of space to store the feature keys along with the other metadata maintained by the package management system is affordable as it is few bytes for each installed feature. In any case, these phases take at most few seconds.

8.1.5 Formula Size in the Protocol with Fully-Hidden Policy

The same evaluation has been executed for the space required to store policies in DNF. In this case, the formula cannot be stored in a compact way since each clause in the DNF has to be written in the installation policy. The expansion factor, defined by $|\phi|$, has been computed for each package in each target distribution. Table 3 contains the cumulative percentage for the distribution of the expansion factors. From these results, it is clear that the additional security requirement of fully hiding the installation policy comes to the cost of a blow-up of the space needed to store the installation policy, and, consequently, of the time needed to share/recovery encryption keys that, in some cases, may be relevant.

8.1.6 Performance on Core Installations

The results reported in the previous paragraphs have been obtained by considering the installation of *all the packages* in

the database. Clearly, in highly sensitive systems, it is never the case that the administrator installs unused and potentially dangerous packages. For this reason we have considered, for each distribution, the set of packages that identify the *core* installation for a *single* distribution version. A *core* installation is the minimal sets of packages for which (a) all dependencies are met, i.e., there exist no dependency from a feature provided by a package outside the set and (b) the installed system is fully functional. Notice that the CentOS *minimal* distribution strictly contains its core installation, as it includes packages that ease the usage of the system but that are not required for the system to work properly (e.g., the X/Windows environment and the LibreOffice productivity suite). Table 1 reports the number of package, provided features and required features for core installations of the distributions in the test set. Such values are one or two orders of magnitude smaller than the values corresponding to the full distributions case. Tables 2 and 3 report the formula size for the public/partially hidden case and the fully hidden one.

8.1.7 Key Processing Time

In this section we briefly report the simulation of key reconstruction algorithms presented in this paper. Specifically, we have only computed the time for (the appropriate number of) share decryptions while we have not reported the time required to query the local inventory, to iterate over keys in therein contained or to run the secret sharing reconstruction algorithm.

We start by defining a set of parameters, $|Z|$, k , c and k_i . As a first step we generate a database of random feature keys of size $|Z|$. We then compute the proper number of shares, $|\phi|$ or $|\hat{\phi}|$, that correspond to an installation policy with parameters k , c and $k_1 = \dots = k_c = k_i$, and encrypt them with the corresponding keys. Given this set of encrypted shares, we run the reconstruction algorithms presented in the paper.

TABLE 3
Cumulative Percentage for the Expansion Factor for the Installation Policy Size in the Fully-Hidden Policies for Full and Core Distributions

Exp. Acronym	Full Distributions				Core Distributions			
	0-10 ³	0-10 ⁴	0-10 ⁵	0-10 ⁶	0-10 ³	0-10 ⁴	0-10 ⁵	0-10 ⁶
CentOS-M	95,64%	98,95%	99,83%	99,83%	95,24%	98,88%	99,72%	100,00%
CentOS-C	84,88%	91,22%	94,50%	96,25%	91,12%	97,63%	98,22%	99,41%
Fedora	87,27%	93,34%	96,19%	97,38%	89,66%	97,24%	97,24%	100,00%
OpenSuse	87,44%	92,99%	96,03%	97,63%	80,90%	90,84%	95,58%	96,76%
RHEL	82,98%	89,78%	93,37%	95,46%	98,19%	98,80%	99,40%	100,00%

TABLE 4
Public and Partially Hidden Policy Key Reconstruction Times

Parameters			Running time (ms)	
$ \mathcal{I} $	k	$ \hat{\phi} $	Public	Part. hidden
1,000	10	50	2	544
1,000	100	140	10	1,130
10,000	10	50	2	2,254
10,000	100	140	11	7,437
100,000	10	50	2	17,340
100,000	100	140	9	70,462

TABLE 5
Fully Hidden Policy Key Reconstruction Times

Parameters			Running time (ms)			
$ \mathcal{I} $	c	k_i	$k = 10$		$k = 50$	
			$ \phi $	time	$ \phi $	time
100	2	5	300	197	1,300	405
100	2	10	1200	367	5,200	667
100	4	5	8,750	1,145	33,750	2,791
100	4	10	140,000	12,282	540,000	40,060
1,000	2	5	300	751	1,300	1,652
1,000	2	10	1,200	1,722	5,200	4,523
1,000	4	5	8,750	8,646	33,750	26,438
1,000	4	10	140,000	126,112	540,000	385,962

Table 4 reports the reconstruction time required by protocols for public or partially hidden policies. We have chosen to test policy sizes $\hat{\phi}$ of size approximately 50 and 100 as these are the limits of the cumulative percentage reported in Table 2. Furthermore, we have chosen the number of keys in the local inventory in the range 10^3 , 10^5 since these are the orders of magnitude for inventories reported in Table 1. Clearly, the time needed in case of public policies does not depend on $|\mathcal{I}|$ since, in this case, given the feature name, the algorithm queries the database and retrieves the correct key. In all cases, few milliseconds are sufficient to correctly decrypt all the shares in the policy. In the case of partially hidden policies, instead, the algorithm needs to iterate over the keys in the inventory. We have considered the worst case scenario in which (a) automatic dependencies always correspond to the last keys in the iteration (i.e., for each such share, the algorithm executes exactly $|\mathcal{I}|$ decryptions) and (b) each manual dependency is fails $k_i/2$ times, (i.e., a successful share decryption follows $|\mathcal{I}|k_i/2$ unsuccessful ones). Also in this case, the timings reported in Table 4 show that the reconstruction times are affordable even in the case of complex policies and huge local inventory.

Table 5 reports the reconstruction time required by protocols for fully hidden policies. In this case we have focused our attention to the case of core distributions.

9 CONCLUSIONS AND OPEN PROBLEMS

In this paper we have presented protocols for ensuring strong dependency enforcement during the software installation process. Our protocols improve over previous solutions for a number of reasons. In our model, we explicitly consider multiple independent software issuers that are not required to cooperate in any way. The enforcement of dependencies is strong in the sense that the package is

distributed in an encrypted form and it can be decrypted by the user if and only if the device meets the issuer-defined installation policy. Finally, the update of the device's key material is executed locally without any interaction with the issuer, solving issues related to the inefficiency of interactive protocols or profiling of user devices. Furthermore, we have presented variants of our protocols in which the description of dependencies can be kept either partially or fully hidden to an adversary by sacrificing, in part, the efficiency of the solution. We have presented and analyzed the protocols from a theoretical point of view and experimentally evaluated their performance.

Our protocols are based on secret sharing schemes that only allow the management of monotone installation policies. We are currently evaluating techniques to allow the management of non-monotone policies, i.e., the ones that allow the installation of a package only if some feature is *not* installed on the system.

ACKNOWLEDGMENTS

This work has been partially supported by the Italian POR projects “WISCH, Work Into Shaping Campania's Home” and by “Piattaforma di Mobilità Intelligente basata su sistema Multi-Agente”. A preliminary version of this paper appears in [1]. G. Persiano Work done while visiting Google.

REFERENCES

- [1] L. Catuogno, C. Galdi, and G. Persiano, “Guaranteeing dependency enforcement in software updates,” in *Proc. 20th Nordic Conf. Secure IT Syst.*, 2015, pp. 205–212.
- [2] T. Dumitras, S. Kavulya, and P. Narasimhan, “A fault model for upgrades in distributed systems (cmu-pdl-08-115),” Carnegie Mellon University, Tech. Rep., Dec. 2008.
- [3] A. D. Rubin, “Trusted distribution of software over the internet,” in *Proc. Symp. Netw. Distrib. Syst. Security*, 1995, pp. 47–53.
- [4] J. Samuel and J. Cappos, “Package managers still vulnerable: How to protect your systems,” *Login: Usenix Mag.*, vol. 34, no. 1, pp. 7–15, 2009.
- [5] J. Cappos, J. Samuel, S. Baker, and J. H. Hartman, “A look in the mirror: Attacks on package managers,” in *Proc. 15th ACM Conf. Comput. Commun. Security*, 2008, pp. 565–574.
- [6] A. Bellissimo, J. Burgess, and K. Fu, “Secure software updates: Disappointments and new challenges,” in *Proc. 6th USENIX Workshop Hot Topics Security*, 2006, pp. 7–7.
- [7] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, “Survivable key compromise in software update systems,” in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 61–72.
- [8] L. Catuogno, R. Gassirà, M. Masullo, and I. Visconti, “SmartK: Smart cards in operating systems at kernel level,” *Inform. Security Tech. Rep.*, vol. 17, no. 3, pp. 93–104, 2013.
- [9] Microsoft Corp., “Introduction to code signing.” [Online]. Available: <https://msdn.microsoft.com/en-us/library/ms537361.aspx>, Microsoft Development Network, last visited Dec. 2017.
- [10] AA. VV., “Signing your applications,” 2005. [Online]. Available: <http://developer.android.com/tools/publishing/app-signing.html>
- [11] J. Cappos, “Avoiding theoretical optimality to efficiently and privately retrieve security updates,” in *Proc. 17th Int. Conf. Financial Cryptography Data Security*, 2013, pp. 386–394.
- [12] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, “Private information retrieval,” *J. ACM*, vol. 45, no. 6, pp. 965–981, Nov. 1998.
- [13] E. Dolstra, M. De Jonge, and E. Visser, “Nix: A safe and policy-free system for software deployment,” in *Proc. 18th USENIX Conf. Syst. Administration*, 2004, vol. 4, pp. 79–92.
- [14] E. Foster-Johnson, *Red Hat RPM Guide*. Wiley & Sons, 2003.
- [15] J. Hart and J. D’Amelia, “An analysis of RPM validation drift,” in *Proc. 18th USENIX Conf. Syst. Administration*, 2002, vol. 2, pp. 155–166.
- [16] S. Neuhaus and T. Zimmermann, “The beauty and the beast: Vulnerabilities in Red Hat’s packages,” in *Proc. USENIX Annu. Tech. Conf.*, 2009, pp. 383–396.

- [17] M. Ambrosin, C. Busold, M. Conti, A.-R. Sadeghi, and M. Schunter, "Updaticator: Updating billions of devices by an efficient, scalable and secure software update distribution over untrusted cache-enabled networks," in *Proc. 16th Eur. Conf. Res. Comput. Security*, 2014, pp. 76–93.
- [18] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.
- [19] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology*. Berlin Heidelberg, Germany: Springer, 2005, vol. 3494, pp. 457–473. [Online]. Available: http://dx.doi.org/10.1007/11426639_27
- [20] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 195–203.
- [21] S. Müller, S. Katzenbeisser, and C. Eckert, *Distributed Attribute-Based Encryption*. Berlin, Heidelberg, Germany: Springer, 2009, pp. 20–36. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00730-9_2
- [22] M. Chase, "Multi-authority attribute based encryption," in *Proc. 4th Conf. Theory Cryptography*, 2007, pp. 515–534.
- [23] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. Au, "PPDCP-ABE: privacy-preserving decentralized ciphertext-policy attribute-based encryption," in *Proc. 19th Eur. Symp. Res. Comput. Security*, 2014, pp. 73–90.
- [24] GlobalPlatform, "Tee system architecture v1.0," Dec. 2011. [Online]. Available: <http://www.globalplatform.org>
- [25] L. Catuogno and C. Galdi, "Ensuring application integrity: A survey on techniques and tools," in *Proc. 9th Int. Conf. Innovative Mobile Internet Serv. Ubiquitous Comput.*, Jul. 2015, pp. 192–199.
- [26] R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proc. USENIX Security Symp.*, 2004, vol. 13, pp. 223–238.
- [27] UEFI, "Unified extensible firmware interface specification," May 2017, pp. 1967–2000. [Online]. Available: http://www.uefi.org/sites/default/files/resources/UEFI_Spec_2_7.pdf
- [28] GlobalPlatform Inc., "TEE System Architecture (version 1.1)," Jan. 2017, <http://www.globalplatform.org>
- [29] J. C. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *Proc. 8th Annu. Int. Cryptology Conf.*, 1988, pp. 27–35.
- [30] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
- [31] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing scheme realizing general access structure," *Electron. Commun.*, vol. 72, no. 9, pp. 56–64, 1989.
- [32] J. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *Proceedings on Advances in Cryptology*. New York, NY, USA: Springer-Verlag, 1990, pp. 27–35.
- [33] D. R. Stinson, "New general lower bounds on the information rate of secret sharing schemes," in *Advances in Cryptology*. Berlin, Germany: Springer, 1993, pp. 168–182.
- [34] G. D. Crescenzo and C. Galdi, "Hypergraph decomposition and secret sharing," *Discrete Appl. Mathematics*, vol. 157, no. 5, pp. 928–946, 2009.
- [35] Common Criteria Sponsoring Organizations, "Common criteria for information technology security evaluation part 1: Introduction and general model, version 3.1 rev 4," Sep. 2012, <http://www.commoncriteriaportal.org/>



Luigi Catuogno received the laurea (MSc equivalent) degree in network administrator from the Department of Computer Science, the University of Salerno, the PhD degree in computer science from the Department of Computer Science, the University of Salerno, and the post-doc fellowship degree from the Department of Computer Science, the University of Salerno, in 1999, 2004, and 2009 respectively. He has been visiting student with New York University, in 2002 and research assistant with the Ruhr University of Bochum (Germany), in 2009. His research focuses on system security and privacy enhancing technologies.



Clemente Galdi received the Laurea cum laude and the PhD degrees in computer science from the Università degli Studi di Salerno, in 1997 and 2002. He has been visiting researcher with Telcordia Technologies and DIMACS, New Jersey. He has been post-doctoral fellow in the Computer Technology Institute and University of Patras (Greece) (2001-2004), Università di Salerno (2004-2006). Since April 2006, he is assistant professor with the Università di Napoli. His fields of interests include the data security, cryptography and algorithm design.



Giuseppe Persiano received the Laurea degree in scienze dell'Informazione from the Università di Salerno, and the PhD degree in computer science from the Harvard University (Cambridge, MA), in 1986 and 1994. He was associate professor with the Università di Catania (1992-1994) and, since 1994 he is professor of computer science with the Università di Salerno. He has visited several research institutions and universities in (DIMACS, Univ. of California at Berkeley, Univ. of California at Los Angeles, Google) and Europe (Univ. of Paderborn, Univ. of Patras). His research focuses on Cryptography and algorithmic aspects of Game Theory and Microeconomic.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

Secure Remote User Authenticated Key Establishment Protocol for Smart Home Environment

Mohammad Wazid¹, Student Member, IEEE, Ashok Kumar Das², Member, IEEE, Vanga Odelu³, Neeraj Kumar⁴, Senior Member, IEEE, and Willy Susilo⁵, Senior Member, IEEE

Abstract—The Information and Communication Technology (ICT) has been used in wide range of applications, such as smart living, smart health and smart transportation. Among all these applications, smart home is most popular, in which the users/residents can control the operations of the various smart sensor devices from remote sites also. However, the smart devices and users communicate over an insecure communication channel, i.e., the Internet. There may be the possibility of various types of attacks, such as smart device capture attack, user, gateway node and smart device impersonation attacks and privileged-insider attack on a smart home network. An illegal user, in this case, can gain access over data sent by the smart devices. Most of the existing schemes reported in the literature for the remote user authentication in smart home environment are not secure with respect to the above specified attacks. Thus, there is need to design a secure remote user authentication scheme for a smart home network so that only authorized users can gain access to the smart devices. To mitigate the aforementioned issues, in this paper, we propose a new secure remote user authentication scheme for a smart home environment. The proposed scheme is efficient for resource-constrained smart devices with limited resources as it uses only one-way hash functions, bitwise XOR operations and symmetric encryptions/decryptions. The security of the scheme is proved using the rigorous formal security analysis under the widely-accepted Real-Or-Random (ROR) model. Moreover, the rigorous informal security analysis and formal security verification using the broadly-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool is also done. Finally, the practical demonstration of the proposed scheme is also performed using the widely-accepted NS-2 simulation.

Index Terms—Smart home, user authentication, key agreement, provable security, AVISPA, NS2 simulation

1 INTRODUCTION

THE advancement of ICT and the Internet have provided the support for rapid growth in smart home environments. A smart home contains the advanced automation systems for monitoring and controlling of various smart devices. In a smart home, the residents can control various smart sensing devices such as temperature monitoring sensors, lighting equipments sensors, or occupancy sensors, etc. [1], [2], [3], [4]. The smart home environment provides a high level of comfort with reduced operational costs to provide safety and security to its residents [5]. One of the major advantages of this type of environment

is for the elderly and disabled people in which these people get assistance in estimating their body parameters using smart gadgets [6]. A smart home is equipped with a number of smart devices (SD_j s), such as low-cost sensors, smart light controllers, smart window shutters, smart AC controllers various and surveillance cameras. Most of the SD_j s are resource-constrained having limited computational and communication power, and limited battery backup [5]. A smart home network can be implemented with the help of these SD_j s in which all SD_j s communicate over wireless channels using the home gateway node (GWN). The GWN acts as a bridge between SD_j s and smart home user (U_i). The GWN provides interoperability and control for the SD_j s and connects them to the external world using the Internet. This facilitates the U_i s to operate the smart home appliances remotely using the Internet-enabled smartphones, tablets, etc. anytime from anywhere in the world [5], [7].

1.1 Network Model

The network model depicted in Fig. 1 consists of the smart home users U_i s who want to access smart devices SD_j s as per their requirements. Suppose there is a user U_i , who wants to access certain SD_j (e.g., temperature & humidity sensor). To access that SD_j , U_i first needs to register himself/herself at the trusted registration authority RA .

- M. Wazid and A. K. Das are with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India. E-mail: mohammad.wazid@research.iiit.ac.in, iitkgp.akdas@gmail.com.
- V. Odelu is with the Department of Computer Science and Engineering, Indian Institute of Information Technology, Sri City, Chittoor 517 588, Andhra Pradesh, India. E-mail: odelu.vanga@gmail.com.
- N. Kumar is with the Department of Computer Science and Engineering, Thapar University, Patiala 147004, India. E-mail: neeraj.kumar@thapar.edu.
- W. Susilo is with the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2500, Australia. E-mail: wsusilo@uow.edu.au.

Manuscript received 14 Nov. 2016; revised 13 Aug. 2017; accepted 10 Oct. 2017. Date of publication 18 Oct. 2017; date of current version 18 Mar. 2020. (Corresponding author: Neeraj Kumar.)
Digital Object Identifier no. 10.1109/TDSC.2017.2764083

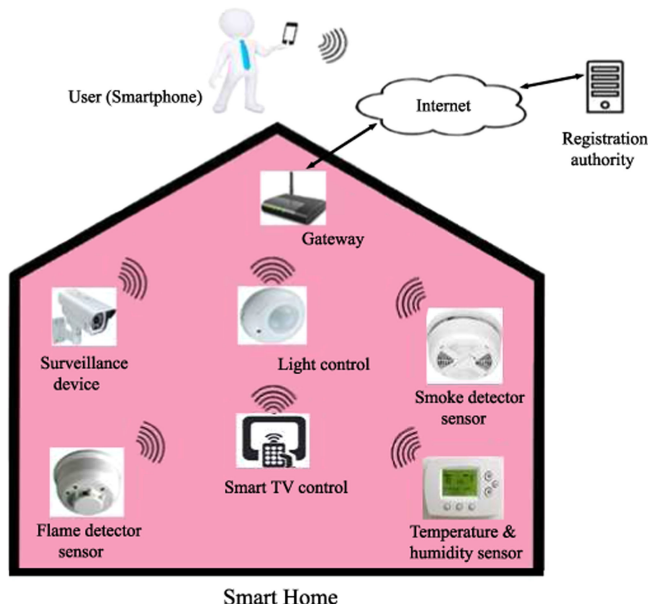


Fig. 1. Smart home environment (Adapted from [5]).

Similarly, all SD_j s and the gateway node GWN (which acts as the bridge between the SD_j and U_i , and connects SD_j to the external world using the Internet) are also registered at the RA . The GWN is thus a special node that takes responsibility of controlling the network data, device and network interoperability and secure management [5]. The registration authority (RA) is a trusted server and it is responsible for registering all the smart devices, users U_i 's and the GWN securely. After the successful registration of U_i , SD_j and GWN securely, the RA stores this useful information in the memory of smart phone SP_i of U_i , and also in the memory of SD_j and GWN , which are further used at the time of authentication and key establishment process. U_i , who wants to access a SD_j , sends an authentication request directly to the GWN as both of them have already performed the registration phase at the RA . Three categories of mutual authentications happen: 1) between U_i and GWN , 2) between GWN and SD_j and 3) between U_i and SD_j . Moreover, U_i and SD_j establish a secret session key SK_{ij} between them to protect the exchanged messages.

1.2 Motivation

Consider the following scenario in smart home environment [8]. Recently, it is noticed that the major trend throughout Europe is the aging society, which is affected by an increasing life expectancy and decreasing birth rates. A large proportion of the European society will be not only from the group of people over 65, but also from a significant increase in the number of people over 80. The proportion of population aged over 65 and over is rising in all countries, however differences can be observed. It is also reported that "the ratio for Iceland, Ireland, Slovak Republic and Turkey lie well below the average for Europe, whereas the ratio for Finland, Germany, Greece, Italy and Sweden lie far above the average for Europe" [8].

The SD_j s in smart homes communicate over the insecure communication channels. There might be the possibility of various attacks in a smart home network. An illegal user (attacker), who can monitor the activities in a smart home,

can break the security, and also can gain access over the SD_j s and other smart home appliances. For example, the attacker can watch the activities in the home by accessing the surveillance camera illegally where disabled people live in the smart home. Most of the existing authentication schemes reported in the literature in a smart home environment are not secure against various known attacks, such as smart device capture attack, user, gateway node and smart device impersonation attacks, and privileged-insider attack. Most of those schemes also fail to preserve traceability and anonymity properties of the users, the GWN as well as of the smart devices SD_j s. Moreover, using the smart phone stolen attack, it is possible that an adversary A can capture a user's secret credentials, such as identity, password and biometrics key with the help of the extracted information stored in the smart phone. In addition, with the help of the user, gateway node and smart device impersonation attacks, A can create valid messages on the behalf of a user U_i , GWN and smart device SD_j , respectively, and can send the corresponding messages to U_i , GWN and SD_j so that these messages are treated as valid by U_i , GWN and SD_j , respectively. In a privileged-insider attack, an insider user of the RA can act as an adversary. The privileged-insider of the RA being an adversary can use the registration information of the users sent to the RA by a legal U_i during the registration phase and derive user's secret credentials, such as identity, password and biometrics key. However, the GWN registration is usually performed in offline mode securely by the RA , and hence, an adversary can not compromise the sensitive information stored in the tamper-resistant GWN device. Considering various possible attacks in a smart home environment, there is a great need to design a secure remote user authentication scheme suitable for a smart home network so that only authorized users can access the information collected by the deployed SD_j s.

1.3 Threat Model

- We have used the Dolev-Yao threat model [9] in our scheme. According to this model, any two communicating parties communicate over an insecure channel and the end-point entities such as U_i and SD_j are not considered as trusted entities. An adversary, say A , can eavesdrop the exchanged messages, and also can modify or delete the message contents during transmission.
- It is assumed that an adversary can physically capture some smart devices equipped at the smart home which are not tamper-resistant, and can extract all the sensitive data stored in those devices.
- As in [5], we also assume that the GWN is fully trusted and can not be compromised by an adversary. Otherwise, the whole network is compromised if the GWN is compromised. For this purpose, as in Bertino et al.'s scheme [10], we also assume that the GWN is equipped with the tamper-resistant device so that all the sensitive information including the cryptographic keying materials stored in it is protected from A . Hence, the use of a tamper-resistant GWN makes the security of the proposed scheme is strong enough. Though the attacks on tamper-resistant devices are possible, the attacker A needs a special equipment to

perform attacks to extract the information. Since it is cheaper to install the *GWN* than the special equipment, so \mathcal{A} does not have economic incentives to mount such an attack [10]. Moreover, the *GWN* can be physically secured by putting it under a locking system inside the smart home of a user so that the physical capture of the *GWN* can be much difficult as compared to that for the smart devices.

- The *RA* is also fully trusted and can not be compromised by an adversary.

1.4 Contributions

Based upon the above discussion, the following contributions are presented in this paper:

- We propose a new remote user authentication scheme for securing a smart home network. The proposed scheme allows three types of mutual authentications: 1) between a user U_i and the *GWN*, 2) between the *GWN* and a smart device SD_j , and 3) a user U_i and a smart device SD_j . At the end, a symmetric session key is established between U_i and SD_j , and they can use the established symmetric key for their future secure communications using a symmetric cipher (for example, the stateless CBC (Cipher Block Chaining) mode of the Advanced Encryption Standard (AES-128), known as AES-CBC [11], [12], [13]).
- The proposed scheme is suitable and efficient for resource-constrained SD_j s with limited resources as it uses only hash invocations, simple bitwise XOR operations and symmetric encryption/decryption operations.
- The security of the proposed scheme is proved using the formal security analysis under the widely-accepted ROR model [14], and also using the rigorous informal security analysis. The formal security discussed in Section 5.1 proves the semantic security of the proposed scheme against an adversary to get the session key between a user and a smart device in the smart home environment. On the other hand, using the informal security analysis, we have shown that the proposed scheme is secure against other possible known attacks, which are discussed in detail in Section 5.3.
- The formal security verification of the proposed scheme in Section 5.2 is done using the broadly-used AVISPA tool [15] and the simulation results show that it is also secure against replay and man-in-the-middle attacks.
- Finally, the practical demonstration of the proposed scheme is provided through the widely-accepted NS-2 simulation [16].

1.5 Roadmap of the Paper

The rest of the paper is structured as follows. We briefly discuss the relevant mathematical preliminaries in Section 2. A brief survey of various existing schemes proposed in the literature is given in Section 3. A new user authentication and session key agreement scheme for smart home environment is presented in Section 4. The rigorous formal and informal security analysis are given in Section 5. In addition, the

formal security verification using the popular AVISPA tool is also given in this section. The practical demonstration of the proposed scheme using widely-accepted NS-2 simulation is given in Section 6. The performance comparison with the existing relevant schemes is given in Section 7. Finally, Section 8 concludes the article.

2 MATHEMATICAL PRELIMINARIES

In this section, we briefly discuss the one-way cryptographic hash function and its properties, and also the indistinguishability of encryption under chosen plaintext attack (IND-CPA), which are necessary to analyze the security of the proposed scheme.

2.1 One-Way Cryptographic Hash Function

A one-way cryptographic hash function $h: \{0, 1\}^* \rightarrow \{0, 1\}^l$ takes an arbitrary-length input, say $x \in \{0, 1\}^*$, and outputs a fixed-length (say, l -bits) message digest $h(x) \in \{0, 1\}^l$.

Definition 1. As defined in [17], the formalization of an adversary \mathcal{A} 's advantage in finding hash collision is given by $Adv_{\mathcal{A}}^{HASH}(t) = \Pr[(a, b) \leftarrow_R \mathcal{A}: a \neq b \text{ and } h(a) = h(b)]$, where $\Pr[X]$ denotes the probability of an event X , and $(a, b) \leftarrow_R \mathcal{A}$ denotes the pair (a, b) is randomly selected by \mathcal{A} . In this case, \mathcal{A} is allowed to be probabilistic and the probability in the advantage is computed over the random choices made by \mathcal{A} with the execution time t . By an (ϵ, t) -adversary \mathcal{A} attacking the collision resistance of $h(\cdot)$, it is meant that the runtime of \mathcal{A} is at most t and that $Adv_{\mathcal{A}}^{HASH}(t) \leq \epsilon$.

2.2 Indistinguishability of Encryption Under Chosen Plaintext Attack

The indistinguishability of encryption under chosen plaintext attack (IND-CPA) is formally defined as follows [18], [19]:

Definition 2. Let *SE/ME* be the single/multiple eavesdropper respectively, and $OR_{ek_1}, OR_{ek_2}, \dots, OR_{ek_N}$ be N different independent encryption oracles associated with encryption keys ek_1, ek_2, \dots, ek_N , respectively. The advantage functions of *SE* and *ME* are defined, respectively, as $Adv_{\Omega, SE}^{IND-CPA}(k) = |2\Pr[SE \leftarrow OR_{ek_1}; (p_0, p_1 \leftarrow_R SE); \delta \leftarrow_R \{0, 1\}; \beta \leftarrow_R OR_{ek_1}(p_\delta) : SE(\beta) = \delta] - 1|$, and $Adv_{\Omega, ME}^{IND-CPA}(k) = |2\Pr[ME \leftarrow OR_{ek_1} \dots OR_{ek_N}; (p_0, p_1 \leftarrow_R ME); \delta \leftarrow_R \{0, 1\}; \beta_1 \leftarrow_R OR_{ek_1}(p_\delta), \dots, \beta_N \leftarrow_R OR_{ek_N}(p_\delta) : ME(\beta_1, \dots, \beta_N) = \delta] - 1|$, where Ω is the encryption scheme. We call Ω is IND-CPA secure in the single (multiple) eavesdropper setting if $Adv_{\Omega, SE}^{IND-CPA}(k)$ (respectively, $Adv_{\Omega, ME}^{IND-CPA}(k)$) is negligible (in the security parameter k) for any probabilistic, polynomial time adversary *SE* (*ME*).

A deterministic encryption scheme means the same message, when it is encrypted twice, yields the same ciphertext. Thus, any deterministic encryption scheme is not IND-CPA secure [13]. There are five modes of symmetric encryption: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter (CTR). Out of these modes, ECB is not IND-CPA secure [13]. Since the adversary knows the Initialization Vector (*IV*), CBC is essentially reduced to ECB, and hence, the stateful CBC is IND-CPA insecure [13]. On the other hand, in the stateless CBC, the *IV* value is chosen at random for each message, and due to this property, the stateless

CBC is IND-CPA secure [13]. If the stateless CBC of AES-128 symmetric encryption scheme is used for encryption/decryption purpose, it then becomes IND-CPA secure.

3 RELATED WORK

Jeong et al. [20] presented a one-time password based user authentication scheme using smart card for smart home networks. Their scheme is lightweight as it uses one-way hash function operations. Their scheme does not provide mutual authentication between *GWN* and smart device as well as between user and smart device. Their scheme does not provide traceability, and user anonymity properties as the user identity is sent in plaintext and also the messages can be easily traced by an adversary. Furthermore, their scheme is insecure against stolen smart card attack and privileged-insider attack as the adversary can derive secret credentials of a user from the extracted information stored in the smart card. In addition, their scheme is not resilient against smart device physical capture attack.

Vaidya et al. [21] proposed a password based remote user authentication scheme for digital home network. Their scheme is also based upon lightweight computation modules such as hashed one-time password and hash-chaining methods. Similar to Jeong et al. [20], their scheme does not provide mutual authentication between *GWN* and smart device as well as between user and smart device. Kim and Kim [22] analyzed Vaidya et al.'s scheme [21] and identified that it is vulnerable to password guessing attack and does not provide forward secrecy with lost smart card. They also proposed a new scheme which withstands the security weaknesses observed in Vaidya et al.'s scheme [21]. Vaidya et al.'s scheme [21] is insecure against stolen smart card attack and privileged-insider attack as the adversary can derive secret credentials of a user from the extracted information stored in the smart card. In addition, their scheme is not resilient against smart device physical capture attack. Later, Vaidya et al. [23] also proposed an elliptic curve cryptography (ECC) based device authentication technique for smart energy home area network which requires more overheads as compared to the scheme in [21]. Kim-Kim's scheme [22] is however not resilient against privileged-insider attack, user impersonation attack and password guessing attack. In addition, Kim-Kim's scheme [22] also fails to preserve traceability and anonymity of user and smart device.

Hanumanthappa et al. [24] proposed a secure three-way authentication mechanism for user authentication and privacy preservation. In their mechanism, the users or service providers can check whether the device is compromised or not by the help of their proposed encrypted pass-phrases mechanism.

Santoso and Vun [25] proposed ECC based user authentication scheme for a smart home system. In their scheme, the mobile user can authenticate with the devices deployed in the smart home using a central node, called the home gateway. Similar to the schemes of Jeong et al. [20], Vaidya et al. [21], and Kim and Kim [22], their scheme does not provide traceability, and user anonymity properties. Furthermore, their scheme is insecure against stolen smart card attack

and privileged-insider attack. In addition, their scheme is not resilient against smart device physical capture attack.

Chang and Le [26] recently proposed a two-factor user authentication scheme in wireless sensor networks (WSNs), which uses a user's password and smart card. Their scheme has two protocols: \mathcal{P}_1 and \mathcal{P}_2 . While \mathcal{P}_1 is based on bitwise XOR and hash functions, \mathcal{P}_2 uses ECC along with bitwise XOR and hash functions. However, Das et al. [27] proved that both \mathcal{P}_1 and \mathcal{P}_2 are insecure against session specific temporary information attack and offline password guessing attack, while \mathcal{P}_1 is also insecure against session key breach attack. Moreover, they pointed out that both \mathcal{P}_1 and \mathcal{P}_2 are inefficient in authentication and password change phases. To erase the security limitations in \mathcal{P}_1 and \mathcal{P}_2 , a new authentication and key agreement scheme using ECC in WSNs is presented [27].

Kumar et al. [5] presented a lightweight and secure session key establishment scheme for smart home network. To establish the mutual trust, each smart device control unit establishes a session key with the *GWN* by using a short authentication token. However, their scheme does not preserve the *GWN* anonymity and also the traceability properties. In addition, their scheme does not provide mutual authentication between user and smart device as well as between user and the *GWN*.

Li et al. [28] proposed an ECC based key establishment scheme for smart home energy management systems. Through the implementation, it is shown that their scheme is efficient with respect to execution time and memory usage. Han et al. [29] presented a secure key agreement scheme for ubiquitous smart home systems, which is particularly applicable to the consumer electronics devices in a smart home. The security and functionality features of the existing schemes summarized in Table 4 are also discussed in detail in Section 7.

4 THE PROPOSED SCHEME

We propose a new user authenticated key establishment scheme for the smart home environment. In the proposed scheme, we have a registration authority, several smart sensing devices, a gateway node (*GWN*) and several users, who want to access the smart devices. First of all, the secure offline registration of each smart device and *GWN* is done at the registration authority (*RA*). Then a user, who wants to access the smart devices, needs to register at the registration authority providing his/her necessary information. Each user has a smart phone, which is capable to read the credential information such as the user's identity, password and biometric (fingerprint scanning etc.) provided by that user. The *GWN* acts as an intermediary node. The legal user's authentication request goes to the *GWN* and then the *GWN* forwards the request to the requested smart device. The smart device sends response to the *GWN* accordingly and then the *GWN* forwards the response to the user. As discussed in the threat model provided in Section 1.3, the *GWN* is fully trusted and all the sensitive informations stored in the *GWN* are protected from an adversary [5]. Moreover, we assume that all the heterogeneous devices (i.e., *GWN*, users (smart phones) and smart devices) are synchronized with their clocks, and agree (mutually) on a

TABLE 1
Notations Used

Notation	Description
RA	Registration authority
GWN	Gateway node
SD_j	j^{th} smart device in the home
U_i	i^{th} user
SP_i	U_i 's smart phone
ID_i	U_i 's identity
ID_{SD_j}	SD_j 's identity
PW_i, BIO_i	U_i 's password & personal biometrics, respectively
T_i	Current timestamp
ΔT	Maximum transmission delay
K_{GWN-U_i}	Secret key of GWN for U_i
K_{GWN-SD_j}	Secret key of GWN for SD_j
$E_K(\cdot)/D_K(\cdot)$	Symmetric encryption/decryption (for example, AES-CBC (128 bits) [12]) using key K
σ_i	Biometric secret key of U_i
τ_i	Public reproduction parameter of U_i
t	Error tolerance threshold used in fuzzy extractor
Gen	Fuzzy extractor probabilistic generation procedure
Rep	Fuzzy extractor deterministic reproduction procedure
$h(\cdot)$	One-way collision-resistant cryptographic hash function
\parallel, \oplus	Concatenation and bitwise XOR operations, respectively

maximum transmission delay (ΔT) to protect replay attacks in the proposed scheme [5].

Our scheme has six phases: 1) offline smart device and gateway registration, 2) user registration, 3) login, 4) authentication and agreement, 5) biometric and password update, and 6) dynamic smart device addition. The notations presented in Table 1 are used in the proposed scheme. We assume that there are m users and n smart devices in the smart home environment. In addition, we assume that n' additional smart devices can be added in the network through the dynamic smart device addition phase, where $n' \ll n$. We also use the fuzzy extractor to verify the biometrics. The fuzzy extractor is a tuple $\langle \mathcal{M}, l, t \rangle$, which is composed of the following two algorithms [30], [31]:

Gen. It is a probabilistic algorithm, which takes a biometric template B_i from a given metric space \mathcal{M} as input, and then outputs a biometric key $\sigma_i \in \{0, 1\}^l$ and a public reproduction parameter τ_i , that is, $Gen(B_i) = \{\sigma_i, \tau_i\}$, where l denotes the number of bits present in σ_i .

Rep. This is a deterministic algorithm, which takes a noisy biometric template $B'_i \in \mathcal{M}$ and a public parameter τ_i and t related to B_i , and then it reproduces (recovers) the biometric key σ_i . In other words, $Rep(B'_i, \tau_i) = \sigma_i$ provided that the Hamming distance between B_i and B'_i is less than or equal to a predefined error tolerance threshold value t .

4.1 Offline Smart Device and Gateway Registration Phase

The offline smart device (SD_j) and GWN registration is done by the registration authority (RA) in offline securely (for example, in person). For each SD_j ($j = 1, 2, \dots, n$), the RA selects a unique identity ID_{SD_j} and also generates a unique random 1024-bit secret key K_{GWN-SD_j} of GWN for SD_j , and computes the corresponding temporal credential $h(ID_{SD_j} \parallel K_{GWN-SD_j})$, and stores $\{ID_{SD_j}, h(ID_{SD_j} \parallel K_{GWN-SD_j})\}$ into the memory of SD_j . The RA further randomly generates the unique GWN 's identity ID_{GWN} and a unique

User (U_i) / Smart phone (SP_i)	Registration authority (RA)
Choose ID_i, PW_i , and imprint BIO_i . Generate 160-bit random secrets a, r . Compute $Gen(BIO_i) = (\sigma_i, \tau_i)$, $RPW_i = h(PW_i \parallel \sigma_i \parallel a) \oplus r$. $\langle ID_i, RPW_i \rangle$ (via a secure channel)	Select 1024-bit K_{GWN-U_i} . Compute $A_i = h(ID_i \parallel K_{GWN-U_i}) \oplus RPW_i$. Generate temporary identity TID_i $\langle A_i, TID_i \rangle$ (via a secure channel)
Compute $B_i = h(ID_i \parallel \sigma_i) \oplus a$, $RPW'_i = RPW_i \oplus r = h(PW_i \parallel \sigma_i \parallel a)$, $C_i = h(ID_i \parallel RPW'_i \parallel \sigma_i)$, $A_i^* = A_i \oplus r$ $= h(ID_i \parallel K_{GWN-U_i}) \oplus RPW'_i$. Delete A_i from SP_i 's memory. Store $\{TID_i, A_i^*, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t\}$ in SP_i 's memory.	corresponding to ID_i . $\langle A_i, TID_i \rangle$ (via a secure channel) Store $\{ID_i, TID_i\}$ in GWN 's database. Delete A_i and RPW_i from its database.

Fig. 2. User registration phase.

random 1024-bit secret key K_{GWN-U_i} of GWN for each user U_i ($i = 1, 2, \dots, m$), and also selects the temporary identity TID_i corresponding to each user U_i 's identity ID_i into the memory of the GWN after U_i 's successful registration phase described in Section 4.2. Finally, the GWN and SD_j contain the information $\{\langle TID_i, ID_i, K_{GWN-U_i} \rangle \mid i = 1, 2, \dots, m\}$, $\{\langle ID_{SD_j}, K_{GWN-SD_j} \rangle \mid j = 1, 2, \dots, n\}$, and $\langle ID_{SD_j}, h(ID_{SD_j} \parallel K_{GWN-SD_j}) \rangle$ for each user U_i and smart device SD_j , respectively.

4.2 User Registration Phase

To access the services from a particular smart device SD_j , a user U_i first needs to register with the RA securely (for example, in person). The following steps are required for the U_i 's registration, which are also summarized in Fig. 2:

Step REG1. U_i chooses a unique identity ID_i and a password PW_i , and generates 160-bit random secrets a and r . U_i also imprints his/her biometrics BIO_i to the sensor of SP_i . The SP_i applies the fuzzy extractor probabilistic generation function $Gen(\cdot)$ to generate secret biometric key σ_i and public parameter τ_i as $Gen(BIO_i) = (\sigma_i, \tau_i)$ [31], [32], [33]. The SP_i of U_i calculates the masked password $RPW_i = h(PW_i \parallel \sigma_i \parallel a) \oplus r$, and sends the registration request $\langle ID_i, RPW_i \rangle$ to the RA using a secure channel. Note that a privileged-insider user of the RA being an adversary knows the registration information $\{ID_i, RPW_i\}$ to mount the privileged-insider attack.

Step REG2. After receiving $\langle ID_i, RPW_i \rangle$ from SP_i , the RA first generates a 1024-bit secret key K_{GWN-U_i} of GWN for U_i , and calculates $A_i = h(ID_i \parallel K_{GWN-U_i}) \oplus RPW_i$. RA also generates a temporary identity TID_i corresponding to ID_i for U_i as discussed in the GWN registration phase (Section 4.1). Finally, RA sends the registration reply with information $\{A_i, TID_i\}$ to U_i securely. Note that the privileged-insider user of the RA being an adversary does not know the information $\{A_i, TID_i\}$ as these information are computed online by the RA .

Step REG3. After receiving $\langle A_i, TID_i \rangle$ from the RA , SP_i of U_i computes parameters $B_i = h(ID_i \parallel \sigma_i) \oplus a$, $RPW'_i = RPW_i \oplus r = h(PW_i \parallel \sigma_i \parallel a)$, $C_i = h(ID_i \parallel RPW'_i \parallel \sigma_i)$ and $A_i^* = A_i \oplus r = h(ID_i \parallel K_{GWN-U_i}) \oplus RPW'_i = h(ID_i \parallel K_{GWN-U_i}) \oplus h(PW_i \parallel \sigma_i \parallel a)$. Finally, SP_i stores the information $\langle TID_i, A_i^*, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t \rangle$ in its memory, where t is the error tolerance parameter used by the fuzzy extractor $Rep(\cdot)$ function.

At the end of this phase, the user U_i erases A_i from his/her smart phone SP_i in order to avoid the privileged-insider attack as explained in Section 5.3.3. In addition, the RA also deletes A_i and RPW_i from its database.

4.3 Login Phase

The login process of U_i is performed as per the following steps:

Step UL1. U_i first provides his/her identity ID_i and password PW_i^* into the interface of the smart phone SP_i , and also provides his/her biometrics BIO_i^* to the sensor of SP_i . SP_i extracts the biometric key σ_i^* as $\sigma_i^* = Rep(BIO_i^*, \tau_i)$ with the constraint that the Hamming distance between the original biometrics BIO_i at the time of registration and entered current BIO_i^* is less than or equal to t . SP_i further computes $a^* = B_i \oplus h(ID_i || \sigma_i^*)$, $RPW_i^* = h(PW_i^* || \sigma_i^* || a^*)$ and $C_i^* = h(ID_i || RPW_i^* || \sigma_i^*)$. SP_i then checks whether $C_i^* = C_i$. If it is valid, U_i passes both password and biometric verification. Otherwise, the session is terminated immediately.

Step UL2. SP_i calculates $M_1 = A_i^* \oplus RPW_i^* = h(ID_i || K_{GWN-U_i})$. Then SP_i generates a random nonce r_{U_i} and the current timestamp T_1 , and calculates parameters $M_2 = M_1 \oplus r_{U_i}$ and $M_3 = h(M_2 || T_1 || ID_i || TID_i || r_{U_i})$. Finally, SP_i sends the login request message $\langle TID_i, M_2, M_3, T_1 \rangle$ to GWN via an open channel.

4.4 Authentication and Key Agreement Phase

On receiving the login request $\langle TID_i, M_2, M_3, T_1 \rangle$ from SP_i , following steps are performed by U_i/SP_i , GWN and an accessed smart device SD_j to establish a session key between U_i and SD_j for later secure communication:

Step AUKA1. GWN first checks the timeliness of T_1 by condition $|T_1 - T_1^*| \leq \Delta T$, where the maximum transmission delay is denoted by ΔT and T_1^* is the reception time of the message $\langle TID_i, M_2, M_3, T_1 \rangle$. If the condition matches, the GWN searches the received TID_i in its database and if it is found in the database, the GWN extracts ID_i and K_{GWN-U_i} corresponding to TID_i from its database, and calculates $M_4 = h(ID_i || K_{GWN-U_i}) (= M_1)$ using the extracted ID_i and K_{GWN-U_i} , $r_{U_i}^* = M_2 \oplus M_4 = M_2 \oplus M_1$, $M_5 = h(M_2 || T_1 || ID_i || TID_i || r_{U_i}^*)$.

Step AUKA2. GWN checks if $M_5 = M_3$ holds. If it does not match, it terminates the authentication process. Otherwise GWN generates a random nonce r_{GWN} and timestamp T_2 , and calculates parameters $M_6 = h(ID_{SD_j} || K_{GWN-SD_j})$, $M_7 = E_{M_6}[ID_i, ID_{GWN}, r_{U_i}^*, r_{GWN}, h(M_4)]$, $M_8 = h(M_6 || T_2 || ID_i || ID_{SD_j} || ID_{GWN} || r_{GWN})$. For computing M_7 , if we use the stateless CBC of AES-128 (AES-CBC) symmetric encryption scheme, then the GWN needs to set the IV of CBC as $IV = h(M_6 || T_1)$ so that it is random for each message in a particular session. Then GWN sends the authentication request message $\langle M_7, M_8, T_2 \rangle$ to SD_j via an open channel.

Step AUKA3. After receiving the message $\langle M_7, M_8, T_2 \rangle$ from GWN , SD_j checks the timeliness of T_2 by the criteria $|T_2 - T_2^*| \leq \Delta T$, where T_2^* is the reception time of the message $\langle M_7, M_8, T_2 \rangle$. If condition holds, SD_j decrypts M_7 using the stored key $h(ID_{SD_j} || K_{GWN-SD_j})$ as $(ID_i, ID_{GWN}, r_{U_i}^*, r_{GWN}, h(M_4)) = D_{h(ID_{SD_j} || K_{GWN-SD_j})}[M_7]$. For decrypting M_7 , SD_j also needs to set the IV of CBC as $IV = h(h(ID_{SD_j} || K_{GWN-SD_j}) || T_1) (= h(M_6 || T_1))$.

Step AUKA4. SD_j calculates $M_9 = h[h(ID_{SD_j} || K_{GWN-SD_j}) || T_2 || ID_i || ID_{SD_j} || ID_{GWN} || r_{GWN}]$ and checks the condition $M_9 = M_8$. If it does not match, it terminates the authentication process. Otherwise, SD_j generates a random nonce r_{SD_j} and the current timestamp T_3 , and computes the session key as $SK_{ij} = h[ID_i || ID_{SD_j} || ID_{GWN} || r_{U_i}^* || r_{GWN} || r_{SD_j}$

$|| h(M_4) || h(h(ID_{SD_j} || K_{GWN-SD_j}))]$. After that, SD_j computes parameters $M_{10} = h(h(ID_{SD_j} || K_{GWN-SD_j}) || T_3) \oplus r_{SD_j}$, $M_{11} = h(SK_{ij} || T_3)$ and $M_{12} = h(r_{SD_j} || r_{GWN} || ID_{SD_j} || ID_{GWN} || T_3)$. Then SD_j sends the authentication reply message $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ to the GWN via an insecure channel.

Step AUKA5. Upon receiving authentication request message, GWN checks the timeliness of T_3 by applying the criteria $|T_3 - T_3^*| \leq \Delta T$, where T_3^* is the reception time of the message $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$. If condition matches, GWN computes $r_{SD_j}^* = M_{10} \oplus h[h(ID_{SD_j} || K_{GWN-SD_j}) || T_3]$ and $M_{13} = h(r_{SD_j}^* || r_{GWN} || ID_{SD_j} || ID_{GWN} || T_3)$. The GWN checks the condition $M_{13} = M_{12}$. If it does not match, the GWN aborts the message. Otherwise, GWN computes M_{14} using previously computed $M_4 = h(ID_i || K_{GWN-U_i})$ as $M_{14} = E_{M_4}[r_{U_i}^*, r_{GWN}, r_{SD_j}^*, ID_{SD_j}, ID_{GWN}, h(M_6)]$. For encrypting the information in M_{14} using the key M_4 , we also use the stateless CBC of AES-128 (AES-CBC) symmetric encryption scheme and thus, the GWN needs to set the IV of CBC as $IV = h(M_4 || T_4)$ so that it is random for each message in a particular session. The GWN chooses current timestamp T_4 and generates a new temporary identity TID_i^{new} corresponding to ID_i . The GWN further computes $M_{15} = TID_i^{new} \oplus h(TID_i || M_4 || T_3 || T_4)$ and $M_{16} = h(M_{11} || T_4 || r_{U_i}^*)$. The GWN sends the message $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ to U_i via insecure channel.

Step AUKA6. After receiving the message $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$, SP_i of U_i first checks the timeliness of T_4 with the condition $|T_4 - T_4^*| \leq \Delta T$, where T_4^* is the reception time of the message. If condition matches, U_i decrypts M_{14} using pre-computed M_1 as $D_{M_1}[M_{14}] = (r_{U_i}^*, r_{GWN}^*, r_{SD_j}^*, ID_{SD_j}, ID_{GWN}, h(M_6))$. For decrypting M_{14} , SD_j also needs to set the IV of CBC as $IV = h(M_1 || T_4) (= h(M_4 || T_4))$.

Then SP_i checks if $r_{U_i}^* = r_{U_i}$. If they do not match, SP_i terminates the authentication process. Otherwise, it computes the session key $SK'_{ij} = h[ID_i || ID_{SD_j} || ID_{GWN} || r_{U_i} || r_{GWN} || r_{SD_j}^* || h(M_1) || h(M_6)]$ and $M_{17} = h(h(SK'_{ij} || T_3) || T_4 || r_{U_i})$, and then matches if $M_{17} = M_{16}$. If it does not match, SP_i terminates the session and discards the computed session key. Otherwise, message comes from the valid source and the computed session key SK'_{ij} is authentic. Finally, SP_i computes the new temporary identity as $TID_i^{new} = M_{15} \oplus h(TID_i || M_1 || T_3 || T_4)$ and replaces TID_i with TID_i^{new} in its memory.

The login, and authentication and agreement phases are summarized in Fig. 3.

4.5 Password and Biometric Update Phase

The proposed scheme provides password and biometric update facility through which a legitimate user U_i can update his/her password and biometrics for security reasons at any time after user registration phase without further involving the RA . Note that the biometric information of a given user U_i is unique and unchanged as compared to the chosen password by that user U_i . However, we suggest the user U_i to update his/her biometric information in the proposed scheme, if he/she desires to do so. This is required to protect strongly the offline password guessing attack to be considered in this phase as described by Huang et al. [34], which is discussed in detail in Section 5.3.11. This phase needs the following steps:

Step PBU1. U_i provides his/her identity ID_i , old password PW_i^{old} to interface of the SP_i and current his/her

User (U_i)/Smart phone (SP_i)	Gateway node (GWN)	Smart device (SD_j)
$\langle TID_i, A_i^*, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t \rangle$	$\{\{TID_i, ID_i, K_{GWN-U_i}\} i = 1, 2, \dots, m\},$ $\{\{ID_{SD_j}, K_{GWN-SD_j}\} j = 1, 2, \dots, n\}$	$\langle ID_{SD_j}, h(ID_{SD_j} K_{GWN-SD_j}) \rangle$
Input ID_i, PW_i^* & BIO_i^* . Compute $\sigma_i^* = Rep(BIO_i^*, \tau_i)$, $a^* = B_i \oplus h(ID_i \sigma_i^*)$, $RPW_i^* = h(PW_i^* \sigma_i^* a^*)$, $C_i^* = h(ID_i RPW_i^* \sigma_i^*)$. Check if $C_i^* = C_i$? If so, compute $M_1 = A_i^* \oplus RPW_i^* = h(ID_i K_{GWN-U_i})$. Generate r_{U_i} & T_1 , and calculate $M_2 = M_1 \oplus r_{U_i}$, $M_3 = h(M_2 T_1 ID_i TID_i r_{U_i})$. $\langle TID_i, M_2, M_3, T_1 \rangle$ (via open channel)	Check if $ T_1 - T_1^* \leq \Delta T$? If so, extract ID_i & K_{GWN-U_i} corresponding to TID_i . Compute $M_4 = h(ID_i K_{GWN-U_i}) (= M_1)$ using extracted ID_i & K_{GWN-U_i} , $r_{U_i}^* = M_2 \oplus M_4$, $M_5 = h(M_2 T_1 ID_i TID_i r_{U_i}^*)$. Check if $M_5 = M_3$? If matches, generate r_{GWN} & T_2 . Compute $M_6 = h(ID_{SD_j} K_{GWN-SD_j})$, $M_7 = E_{M_6}[ID_i, ID_{GWN}, r_{U_i}^*, r_{GWN}, h(M_4)]$, $M_8 = h(M_6 T_2 ID_i ID_{SD_j} ID_{GWN} r_{GWN})$. $\langle M_7, M_8, T_2 \rangle$ (via open channel)	Check if $ T_2 - T_2^* \leq \Delta T$? If so, decrypt M_7 to retrieve $(ID_i, ID_{GWN}, r_{U_i}^*, r_{GWN}, h(M_4))$ $= D_{h(ID_{SD_j} K_{GWN-SD_j})}[M_7]$. Compute $M_9 = h(h(ID_{SD_j} K_{GWN-SD_j})$ $ T_2 ID_i ID_{SD_j} ID_{GWN} r_{GWN})$. Check if $M_9 = M_8$? If so, generate r_{SD_j} & T_3 , and compute $SK_{ij} = h(ID_i ID_{SD_j} $ $ID_{GWN} r_{U_i}^* r_{GWN} r_{SD_j} h(M_4)$ $ h(h(ID_{SD_j} K_{GWN-SD_j})))$, $M_{10} = h(h(ID_{SD_j} K_{GWN-SD_j}) T_3) \oplus r_{SD_j}$, $M_{11} = h(SK_{ij} T_3)$, $M_{12} = h(r_{SD_j} r_{GWN} ID_{SD_j} ID_{GWN} T_3)$. $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ (via open channel)
Check if $ T_4 - T_4^* \leq \Delta T$? If so, decrypt $D_{M_1}[M_{14}] = (r_{U_i}^*, r_{GWN}^*, r_{SD_j}^*,$ $ID_{SD_j}, ID_{GWN}, h(M_6))$. Check if $r_{U_i}^* = r_{U_i}$? If so, compute $SK_{ij}^* = h(ID_i ID_{SD_j}$ $ ID_{GWN} r_{U_i} r_{GWN}^* r_{SD_j}^*$ $ h(M_1) h(M_6))$, $M_{17} = h(h(SK_{ij}^* T_3) T_4 r_{U_i})$. Check if $M_{17} = M_{16}$? If so, U_i and SD_j establish session key $SK_{ij} (= SK_{ij}^*)$. Compute $TID_i^{new} = M_{15} \oplus h(TID_i$ $ M_1 T_3 T_4)$. Replace TID_i with TID_i^{new} .	Check if $ T_3 - T_3^* \leq \Delta T$? If so, compute $r_{SD_j}^* = M_{10} \oplus$ $h(h(ID_{SD_j} K_{GWN-SD_j}) T_3)$, $M_{13} = h(r_{SD_j}^* r_{GWN} ID_{SD_j} ID_{GWN} T_3)$. Check if $M_{13} = M_{12}$? If so, compute $M_{14} = E_{M_4}[r_{U_i}^*, r_{GWN}^*, r_{SD_j}^*,$ $ID_{SD_j}, ID_{GWN}, h(M_6)]$. Generate T_4 , select TID_i^{new} and compute $M_{15} = TID_i^{new} \oplus h(TID_i M_4 T_3 T_4)$, $M_{16} = h(M_{11} T_4 r_{U_i}^*)$. $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ (via open channel)	

Fig. 3. Summary of login, and authentication and key agreement phases.

biometrics BIO_i^{old} to the sensor of the SP_i . SP_i then computes $\sigma_i^{old} = Rep(BIO_i^{old}, \tau_i)$, $a' = B_i \oplus h(ID_i || \sigma_i^{old})$, $RPW_i^{old} = h(PW_i^{old} || \sigma_i^{old} || a')$ and $C_i^{old} = h(ID_i || RPW_i^{old} || \sigma_i^{old})$. SP_i checks the condition $C_i^{old} = C_i$. If it matches, U_i is the actual user; otherwise, the phase is terminated immediately.

Step PBU2. SP_i asks U_i to enter a new password PW_i^{new} and also imprint new biometrics BIO_i^{new} . The SP_i then calculates $Gen(BIO_i^{new}) = (\sigma_i^{new}, \tau_i^{new})$, $RPW_i^{new} = h(PW_i^{new} || \sigma_i^{new} || a')$, $B_i^{new} = h(ID_i || \sigma_i^{new}) \oplus a'$, $C_i^{new} = h(ID_i || RPW_i^{new} || \sigma_i^{new})$ and $A_i^{new} = A_i^* \oplus RPW_i^{old} \oplus RPW_i^{new} = h(ID_i || K_{GWN-U_i}) \oplus RPW_i^{new} = h(ID_i || K_{GWN-U_i}) \oplus h(PW_i^{new} || \sigma_i^{new} || a')$.

Step PBU3. Finally, SP_i replaces τ_i , A_i^* , B_i , and C_i with τ_i^{new} , A_i^{new} , B_i^{new} , and C_i^{new} in its memory, respectively.

The password and biometric update phase is also summarized in Fig. 4.

4.6 Dynamic Smart Device Addition Phase

To deploy a new smart device SD_j^{new} in the existing smart home network, the RA performs the following steps in offline:

Step DA1. RA first assigns a unique new identity $ID_{SD_j}^{new}$ and also generates a new secret key $K_{GWN-SD_j}^{new}$ of GWN for SD_j^{new} . RA further computes the temporal credential of SD_j^{new} as $h(ID_{SD_j}^{new} || K_{GWN-SD_j}^{new})$.

Step DA2. RA stores the information $\{ID_{SD_j}^{new}, h(ID_{SD_j}^{new} || K_{GWN-SD_j}^{new})\}$ into the memory of SD_j before its deployment in the smart home. RA also sends the information $\{ID_{SD_j}^{new}, K_{GWN-SD_j}^{new}\}$ to the GWN securely, which are then stored in the database of the GWN .

Finally, RA also needs to inform the existing users in the network about the deployment of new smart device SD_j^{new} so that they can access the services from SD_j^{new} , if needed.

5 SECURITY ANALYSIS

In this section, we analyze the security of the proposed scheme using both formal and informal analysis.

5.1 Formal Security Analysis Using Real-Or-Random Model

The widely-accepted Real-Or-Random (ROR) model [14] is used for formal security analysis of the proposed scheme.

5.1.1 ROR Model

We follow the Abdalla et al.'s ROR model [14] for formal security analysis as done in [26]. According to our scheme,

User (U_i)	Smart phone (SP_i)
	$\langle TID_i, A_i^*, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t \rangle$
Provide ID_i, PW_i^{old} & BIO_i^{old} .	Compute $\sigma_i^{old} = Rep(BIO_i^{old}, \tau_i)$, $a' = B_i \oplus h(ID_i \sigma_i^{old})$, $RPW_i^{old} = h(PW_i^{old} \sigma_i^{old} a')$, $C_i^{old} = h(ID_i RPW_i^{old} \sigma_i^{old})$. Check if $C_i^{old} = C_i$? If so, ask U_i to provide new password & biometrics.
Provide PW_i^{new} & BIO_i^{new} .	Compute $Gen(BIO_i^{new}) = (\sigma_i^{new}, \tau_i^{new})$, $RPW_i^{new} = h(PW_i^{new} \sigma_i^{new} a')$, $B_i^{new} = h(ID_i \sigma_i^{new}) \oplus a'$, $C_i^{new} = h(ID_i RPW_i^{new} \sigma_i^{new})$, $A_i^{new} = A_i^* \oplus RPW_i^{old} \oplus RPW_i^{new}$, $= h(ID_i K_{GWN-U_i}) \oplus RPW_i^{new}$. Finally, SP_i replaces τ_i , A_i^* , B_i and C_i with τ_i^{new} , A_i^{new} , B_i^{new} and C_i^{new} , respectively.

Fig. 4. Summary of password and biometric update phase.

we have three participants in the smart home: smart device SD_j , user U_i and GWN .

Participants. Let $\Pi_{SD_j}^t$, $\Pi_{U_i}^u$ and Π_{GWN}^v be the instances t , u and v of SD_j , U_i and GWN , respectively. These are called oracles [26].

Accepted State. An instance Π^t is known to be accepted, if upon receiving the last expected protocol message, it goes into an accept state. The ordered concatenation of all communicated (sent and received) messages by Π^t forms the session identification (sid) of Π^t for the current session.

Partnering. Two instances Π^{t_1} and Π^{t_2} are said to be partnered if the following three conditions are fulfilled simultaneously: 1) both Π^{t_1} and Π^{t_2} are in accept state; 2) both Π^{t_1} and Π^{t_2} mutually authenticate each other and share the same sid ; and 3) Π^{t_1} and Π^{t_2} are mutual partners of each other.

Freshness. The instance $\Pi_{U_i}^u$ or $\Pi_{SD_j}^t$ is fresh, if the session key SK_{ij} between U_i and SD_j has not revealed to an adversary \mathcal{A} using the $\text{Reveal}(\Pi^t)$ query given below [26].

Adversary. It is assumed that \mathcal{A} has fully control over all the communications in a smart home. \mathcal{A} has the ability to read, modify the exchanged messages, or can fabricate new messages and inject them into the network. Furthermore, \mathcal{A} has access to the following queries [26]:

Execute(Π^u, Π^v, Π^t): \mathcal{A} can execute this query to obtain the messages exchanged between three legitimate participants U_i , GWN and SD_j , which is further modeled as an eavesdropping attack.

Reveal(Π^t): This query reveals the current session key SK_{ij} generated by Π^t (and its partner) to an adversary \mathcal{A} .

Send(Π^t, msg): \mathcal{A} runs this query to send a message, say msg , to a participant instance Π^t , and also receives a response message. It is modeled as an active attack.

CorruptSmartPhone($\Pi_{U_i}^u$): It represents the smart phone SP_i lost/stolen attack, which outputs the information stored in SP_i .

CorruptSmartDevice($\Pi_{SD_j}^t$): This represents an attack in which secret $h(ID_{SD_j} || K_{GWN-SD_j})$ is disclosed to \mathcal{A} , which is applied to verify the security of the proposed scheme. As mentioned in [26], both *CorruptSmartPhone* and *CorruptSmartDevice* queries ensure the weak-corruption model in which temporary keys and the internal data of the participant instances are not corrupted.

Test(Π^t): It represents the semantic security of a session key SK_{ij} between U_i and SD_j following the indistinguishability in the ROR model [14]. An unbiased coin b is flipped before start of the experiment, and its result is only known to \mathcal{A} which is used to decide the output of the *Test* query. If \mathcal{A} runs this query, and the established session key SK_{ij} is also new, then Π^t returns SK_{ij} in case $b = 1$ or a random number for $b = 0$; otherwise, it outputs \perp (null).

Note that we impose a restriction that the adversary \mathcal{A} has access to only limited number of *CorruptSmartPhone*($\Pi_{U_i}^u$) and *CorruptSmartDevice*($\Pi_{SD_j}^t$) queries, whereas he/she can access the *Test*(Π^t) query many times. According to the threat model described in Section 1.3, the GWN is trusted. Thus, \mathcal{A} does not have any access to a corrupt query related to the GWN .

Semantic Security of Session Key. According to the requirements of the ROR model [14], \mathcal{A} needs to distinguish between an instance's real session key and a random key. \mathcal{A} can make several *Test* queries to either $\Pi_{SD_j}^t$ or $\Pi_{U_i}^u$. The output of *Test* query should be consistent with respect to the random bit b . After the experiment is finished, \mathcal{A} returns a guessed bit b' and he/she can win the game if the condition $b' = b$ is met. Let $SUCC$ be an event that \mathcal{A} win the game. The advantage $Adv_{\mathcal{P}}^{AKE}$ of \mathcal{A} in breaking the semantic security of our authenticated key agreement (AKE) scheme, say \mathcal{P} against deriving the session key SK_{ij} between U_i and SD_j is given by $Adv_{\mathcal{P}}^{AKE} = |2 \cdot Pr[SUCC] - 1|$. In the ROR sense, \mathcal{P} is secure if $Adv_{\mathcal{P}}^{AKE} \leq \psi$, where $\psi > 0$ is a sufficiently small real number.

Random Oracle. As mentioned in [26], all communicating participants as well as \mathcal{A} have access to a collision-resistant one-way cryptographic hash function $h(\cdot)$. $h(\cdot)$ is modeled by a random oracle, say HO .

5.1.2 Security Proof

Theorem 1 provides the semantic security of our proposed scheme under the widely-accepted ROR model [26], [35].

Theorem 1. *Let \mathcal{A} be an adversary running in polynomial time t against our scheme \mathcal{P} in the random oracle, D a uniformly distributed password dictionary and l the number of bits present in the biometrics key σ_i . The advantage of \mathcal{A} in breaking semantic security of our scheme is estimated as $Adv_{\mathcal{P}}^{AKE} \leq$*

$$\frac{q_h^2}{|Hash|} + \frac{q_{send}}{2^{l-1}|D|} + 2Adv_{\Omega}^{IND-CPA}(k), \text{ where } q_h, q_{send}, |Hash|, |D| \text{ and } Adv_{\Omega, SE}^{IND-CPA}(k) \text{ or } Adv_{\Omega, ME}^{IND-CPA}(k) \text{ are the number of HO queries, the Send queries, the range space of } h(\cdot), \text{ the size of } D, \text{ and the advantage of } \mathcal{A} \text{ in breaking the IND-CPA secure symmetric cipher } \Omega \text{ (provided in Definition 2), respectively, and } Adv_{\Omega}^{IND-CPA}(k) = Adv_{\Omega, SE}^{IND-CPA}(k) \text{ or } Adv_{\Omega, ME}^{IND-CPA}(k).$$

Proof. The proof is similar to that presented in the schemes [26], [35]. The sequence of five games, say GM_i , are defined in the security analysis, where $i = 0, 1, 2, 3, 4$. Assume that $SUCC_i$ be an event wherein an adversary \mathcal{A} can guess the random bit b in GM_i correctly.

GM_0 : This game corresponds to a real attack performed by \mathcal{A} against our scheme \mathcal{P} in the ROR sense. The bit b is chosen at the beginning of GM_0 . Hence, it follows that

$$Adv_{\mathcal{P}}^{AKE} = |2 \cdot Pr[SUCC_0] - 1|. \quad (1)$$

GM_1 : This game represents an eavesdropping attack performed by the single/multiple eavesdropper SE/ME , say \mathcal{A} , where \mathcal{A} can query *Execute*(Π^u, Π^v, Π^t) oracle. At the end of the game, \mathcal{A} makes queries to the *Test* oracle. The output of *Test* oracle determines whether it is the actual session key SK_{ij} or a random number. Note that the session key SK_{ij} is calculated by both U_i and SD_j as $SK_{ij} = h(ID_i || ID_{SD_j} || ID_{GWN} || r_{U_i}^* || r_{GWN} || r_{SD_j} || h(M_4) || h(h(ID_{SD_j} || K_{GWN-SD_j})))$, where $M_4 = h(ID_i || K_{GWN-U_i})$. To calculate SK_{ij} , \mathcal{A} must have M_4 and $h(ID_{SD_j} || K_{GWN-SD_j})$, which further involve secret keys K_{GWN-U_i} and K_{GWN-SD_j} . \mathcal{A} also requires ID_i , ID_{SD_j} , ID_{GWN} , r_{U_i} , r_{GWN} and r_{SD_j} for calculating SK_{ij} , which are unknown to him/her. As a consequence, the chance

of winning the game GM_1 for \mathcal{A} is not increased by eavesdropping attack. It is then obvious that

$$\Pr[SUCC_0] = \Pr[SUCC_1]. \quad (2)$$

GM_2 : By adding the simulations of the *Send* and *HO* oracles are added into GM_1 , GM_1 is transformed into GM_2 , which represents an active attack. In this game, the objective of \mathcal{A} is to fool a participant to accept a modified message. \mathcal{A} is permitted to make different *HO* queries to examine the existence of the hash collisions. All the exchanged messages $\langle TID_i, M_2, M_3, T_1 \rangle$, $\langle M_7, M_8, T_2 \rangle$, $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ and $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ during the login and authentication phase contain the participant's identity, random nonce and timestamps. Hence, there is no collision when the *Send* oracle is queried by \mathcal{A} . The results of the birthday paradox give

$$|\Pr[SUCC_1] - \Pr[SUCC_2]| \leq q_h^2 / (2|\text{Hash}|). \quad (3)$$

GM_3 : GM_2 is transformed into GM_3 by adding the simulation of *CorruptSmartPhone* oracle. \mathcal{A} can choose low-entropy passwords, and using the information stored into SP_i he/she may try to acquire the user's password using the dictionary attack. Again, \mathcal{A} may try to acquire the biometrics key σ_i from the information stored in SP_i . We have used a strong fuzzy extractor in our scheme \mathcal{P} , which is capable to extract at most l random bits and the guessing probability of $\sigma_i \in \{0, 1\}^l$ by \mathcal{A} is approximately $\frac{1}{2^l}$ [31]. It is also assumed that the system allows the limited number of wrong password inputs. Thus, we have the following result,

$$|\Pr[SUCC_2] - \Pr[SUCC_3]| \leq q_{send} / (2^l \cdot |D|). \quad (4)$$

GM_4 : GM_3 is transformed into GM_4 , where GM_4 is the last game. It models an attack in which \mathcal{A} can physically capture (compromise) a smart device SD_j by adding the simulation of *CorruptSmartDevice* oracle. \mathcal{A} then knows the information $\{ID_{SD_j}, h(ID_{SD_j} || K_{GWN-SD_j})\}$ which is stored in SD_j . Let \mathcal{A} also has all the eavesdropped messages $\langle TID_i, M_2, M_3, T_1 \rangle$, $\langle M_7, M_8, T_2 \rangle$, $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ and $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$. Then, \mathcal{A} tries to retrieve the information $\{ID_i, ID_{GWN}, r_{U_i}, r_{GWN}, h(M_4)\}$ by decrypting M_7 using $h(ID_{SD_j} || K_{GWN-SD_j})$ as $(ID_i, ID_{GWN}, r_{U_i}^*, r_{GWN}, h(M_4)) = D_{h(ID_{SD_j} || K_{GWN-SD_j})} [M_7]$. However, \mathcal{A} can not decrypt M_{14} as M_4 is unknown to him/her since as $M_{14} = E_{M_4} [r_{U_i}^*, r_{GWN}, r_{SD_j}^*, ID_{SD_j}, ID_{GWN}, h(M_6)]$. This implies that without having $M_4 = h(ID_i || K_{GWN-U_i}) (= M_1)$, it is quite difficult task for \mathcal{A} to extract the information $\{r_{U_i}^*, r_{GWN}, r_{SD_j}^*, ID_{SD_j}, ID_{GWN}, h(M_6)\}$. Thus, computation of the session key $SK_{ij} = h[ID_i || ID_{SD_j} || ID_{GWN} || r_{U_i} || r_{GWN} || r_{SD_j} || h(M_1) || h(M_6)] (= SK'_{ij})$ is difficult as \mathcal{A} needs the necessary information including r_{SD_j} and $M_1 (= M_4)$ due to the IND-CPA secure symmetric cipher used in the proposed scheme for encryption/decryption. This concludes that

$$|\Pr[SUCC_3] - \Pr[SUCC_4]| \leq Adv_{\Omega}^{IND-CPA}(k). \quad (5)$$

In GM_4 , all the random oracles are simulated. \mathcal{A} is only left to guess the bit b for winning the game after querying the *Test* oracle. It is clear that $\Pr[SUCC_4] = 1/2$.

From Equation (1), we get, $\frac{1}{2} \cdot Adv_{\mathcal{P}}^{AKE} = |\Pr[SUCC_0] - \frac{1}{2}|$. Using the triangular inequality, we have, $|\Pr[SUCC_1] - \Pr[SUCC_4]| \leq |\Pr[SUCC_1] - \Pr[SUCC_2]| + |\Pr[SUCC_2] - \Pr[SUCC_4]| \leq |\Pr[SUCC_1] - \Pr[SUCC_2]| + |\Pr[SUCC_2] - \Pr[SUCC_3]| + |\Pr[SUCC_3] - \Pr[SUCC_4]| \leq \frac{q_h^2}{2 \cdot |\text{Hash}|} + \frac{q_{send}}{2^l \cdot |D|} + Adv_{G_q}^{ECDHDP}(t)$.

Using Equations (2) – (5), we have,

$$|\Pr[SUCC_0] - 1/2| \leq q_h^2 / (2 \cdot |\text{Hash}|) + q_{send} / (2^l \cdot |D|) + Adv_{\Omega}^{IND-CPA}(k). \quad (6)$$

Finally, Equation (6) yields the required result:

$$Adv_{\mathcal{P}}^{AKE} \leq \frac{q_h^2}{|\text{Hash}|} + \frac{q_{send}}{2^{l-1} \cdot |D|} + 2Adv_{\Omega}^{IND-CPA}(k). \quad \square$$

5.2 Formal Security Verification Using AVISPA

The proposed scheme is simulated for the formal security verification using the broadly-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool to exhibit that the proposed scheme withstands replay and man-in-the-middle attacks.

AVISPA integrates four back ends that implement different state-of-the-art automatic analysis mechanisms: (i) OFMC; (ii) CL-AtSe; (iii) SATMC; and (iv) TA4SP. The detailed description and functionality of these back ends are available in [15], [35], [36], [37], [38]. A security protocol requires to be implemented in the High Level Protocols Specification Language (HLSL) [39], which is converted into intermediate format (IF) using the HLSL2IF translator. The IF is then given as input to one of the four backends to produce output, which has various sections highlighting whether the designed scheme is safe or unsafe against an adversary.

The registration, login, authentication and session key agreement phases of our scheme are implemented in HLSL. In our implementation, four basic roles are defined: *registration authority*, *user*, *gateway node* and *smart device* for representing the *RA*, a user U_i , the *GWN* and a smart device SD_j , respectively. The HLSL role specification *user* for U_i is given in Fig. 5. U_i as an initiator receives the start signal, updates its state from 0 to 1, and sends the registration request $\langle ID_i, RPW_i \rangle$ to the *RA* using *Snd()* channel securely. The *RA* accepts the registration request of U_i , and sends information $\langle A_i, TID_i \rangle$ to U_i using *Snd()* channel securely. U_i then receives information $\langle A_i, TID_i \rangle$ using *Rcv()* channel securely. U_i sends the login request $\langle TID_i, M_2, M_3, T_1 \rangle$ to the *GWN* using public channel. The *GWN* further sends the authentication request $\langle M_7, M_8, T_2 \rangle$ to SD_j using public channel. The SD_j also sends reply message $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ to the *GWN* using public channel. Finally, the *GWN* sends authentication reply $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ to U_i using public channel. Both *Snd()* and *Rcv()* public channels use Dolev-Yao threat model type [9]. So, an intruder (always denoted by (i)) can read, modify or delete the contents of exchanged messages. Similarly, we also have specified the roles for *RA*, *GWN* and SD_j in our HLSL implementation.

In the session role specified in Fig. 6, all the basic roles are started with concrete arguments. Fig. 6 also consists of

```

role user (Ui, RA, GWN, SDj: agent, H : hash_func,
          SKuira : symmetric_key, Snd, Rcv: channel(dy))
played_by Ui
def=
local State: nat, IDi, IDsdj, IDgwn, PWi, BIOi, RPWi, A: text,
      R, Kgwnui, Kgwnsdj, Rgwn, Rsdj, T1, M1, Rui, TIDi, TIDinew: text,
      M2, M3, T3, T4, Sigmair: text, Gen, Rep : hash_func
const ui_gwn_t1, ui_gwn_rui, gwn_ui_t4, gwn_ui_tidinew.sr1, sr2: protocol_id
init State := 0
transition
1. State = 0  $\wedge$  Rcv(start) =>
% Registration phase
State' := 1  $\wedge$  A' := new()  $\wedge$  R' := new()
 $\wedge$  secret({PWi, BIOi, A', R'}, sr1, Ui)
 $\wedge$  Sigmair' := Gen(BIOi)  $\wedge$  RPWi' := xor(H(PWi.Sigmair'.A'), R')
% Send registration request securely to RA
 $\wedge$  Snd({IDi.RPWi'}_SKuira)
% Receive information securely from RA for SPi
2. State = 1/Rcv({xor(H(IDi.Kgwnui),xor(H(PWi.Sigmair'.A'),R')),TIDi'}_SKuira)}=>
% Login phase
State' := 2  $\wedge$  secret({Kgwnui,Kgwnsdj}, sr2, GWN)
% Send login request to GWN via public channel
 $\wedge$  Rui' := new()  $\wedge$  T1' := new()  $\wedge$  M1' := H(IDi.Kgwnui)
 $\wedge$  M2' := xor(M1', Rui')  $\wedge$  M3' := H(M2'.T1'.IDi'.TIDi'.Rui')
 $\wedge$  Snd(TIDi'.M2'.M3'.T1')
% Ui has freshly generated the values T1 and Rui for GWN
 $\wedge$  witness(Ui,GWN,ui_gwn_t1, T1') $\wedge$ witness(Ui,GWN,ui_gwn_rui,Rui')
% Authentication and key agreement phase
% Receive authentication reply from GWN via public channel
3. State = 2  $\wedge$  Rcv({Rui'.Rgwn'.Rsdj'.IDi.IDsdj.IDgwn.
H(H(IDsdj.Kgwnsdj))_H(IDi.Kgwnui).
xor(TIDinew', H(TIDi'.H(IDi.Kgwnui).T3'.T4')).
H(H(H(IDi.IDsdj.IDgwn.Rui'.Rgwn'.Rsdj'.
H(H(IDi.Kgwnui)).H(H(IDsdj.Kgwnsdj)))_T3'.T4'.Rui').T3'.T4')}=>
% Ui's acceptance of T4 and TIDinew generated for Ui by GWN
State' := 3/request(GWN,Ui,gwn_ui_t4,T4')/request(GWN,Ui,gwn_ui_tidinew,TIDinew')
end role

```

Fig. 5. The user U_i 's role in HLPSSL.

top level *environment* role, which is the starting point for the execution. At the end, in the goal section, four authentication goals and two secrecy goals are specified.

The declaration $witness(U_i, GWN, ui_gwn_t1, T1')$ says that U_i has freshly generated the current timestamp T_1 for GWN . The declaration $request(GWN, U_i, gwn_ui_t4, T4')$ expresses U_i 's acceptance of timestamp T_4 generated for U_i

```

role session (Ui, RA, GWN, SDj: agent, H: hash_func, SKuira: symmetric_key)
def=
local S1, R1, S2, R2, S3, R3, S4, R4: channel (dy)
composition
user (Ui, RA, GWN, SDj, H, SKuira, S1, R1)
 $\wedge$  registrationauthority(Ui, RA, GWN, SDj, H, SKuira, S2, R2)
 $\wedge$  gatewaynode (Ui, RA, GWN, SDj, H, SKuira, S3, R3)
 $\wedge$  smartdevice (Ui, RA, GWN, SDj, H, SKuira, S2, R2)
end role

role environment()
def=
const ui, ra, gwn, sdj: agent, h: hash_func, skuira: symmetric_key,
kgwnui, kgwnsdj, idi, idnsj, idgwn, t1, t2, t3, t4, tidi, tidinew: text,
gen, rep: hash_func, ui_gwn_t1, ui_gwn_rui, gwn_sdj_t2, gwn_sdj_rgwn,
sdj_gwn_t3, sdj_gwn_rsdj, sr1, s2: protocol_id
intruder_knowledge = {t1, t2, t3, t4, h, gen, rep}
composition
session(ui, ra, gwn, sdj, h, skuira)  $\wedge$  session(i, ra, gwn, sdj, h, skuira)
 $\wedge$  session(ui, i, gwn, sdj, h, skuira)  $\wedge$  session(ui, ra, i, sdj, h, skuira)
 $\wedge$  session(ui, ra, gwn, i, h, skuira)
end role

goal
secrecy_of sr1, sr2
authentication_on ui_gwn_t1, ui_gwn_rui, gwn_sdj_t2
authentication_on gwn_sdj_rgwn, sdj_gwn_t3, sdj_gwn_rsdj
authentication_on gwn_ui_t4, gwn_ui_tidinew
end goal
environment()

```

Fig. 6. The session, goal and environment roles in HLPSSL.

% OFMC	SUMMARY
% Version of 2006/02/13	SAFE
SUMMARY	DETAILS
SAFE	BOUNDED_NUMBER_OF_SESSIONS
DETAILS	TYPED_MODEL
BOUNDED_NUMBER_OF_SESSIONS	PROTOCOL
PROTOCOL	C:\progra~1\SPAN\testsuite
C:\progra~1\SPAN\testsuite	\results\user_auth.if
\results\user_auth.if	GOAL
GOAL	As Specified
as_specified	BACKEND
BACKEND	CL-AtSe
OFMC	STATISTICS
COMMENTS	STATISTICS
STATISTICS	Analysed : 8 states
parseTime: 0.00s	Reachable : 0 states
searchTime: 7.75s	Translation: 0.14 seconds
visitedNodes: 1432 nodes	Computation: 0.00 seconds
depth: 8 plies	

Fig. 7. The results of the analysis using OFMC and CL-AtSe backends.

by GWN . The declaration $secret(\{PW_i, A', R'\}, sr1, U_i)$ also says that the information PW_i , a and r are only known to U_i . This is specified with protocol id $sr1$ in the goal section (given in Fig. 6).

We have simulated our scheme using the widely-used OFMC and CL-AtSe backends. The executability check on non-trivial HLPSSL specifications, replay attack check, and Dolev-Yao model check are verified in the proposed scheme. For more details on these verifications, one can refer to [31], [40]. The simulation results shown in Fig. 7 depicts that the proposed scheme is secure against replay as well as man-in-the-middle attacks.

5.3 Informal Security Analysis

The informal security analysis shows that the following other possible known attacks are prevented.

5.3.1 Traceability

In many applications, it is desirable that a user authentication should not allow an adversary to trace a user during login and authentication phases. Therefore, it also becomes important that the identity of the user should not be revealed to an adversary to preserve the privacy of that user in a network, especially in a smart home environment. The login request $\langle TID_i, M_2, M_3, T_1 \rangle$ sent by U_i to the GWN is different each time due to the following reason. The smart phone SP_i of U_i computes $M_1 = A^* \oplus RPW_i^* = h(ID_i || K_{GWN-U_i})$, $M_2 = M_1 \oplus r_{U_i}$ and $M_3 = h(M_2 || T_1 || ID_i || TID_i || r_{U_i})$, where T_1 is current timestamp and r_{U_i} random nonce of U_i . The involvement of T_1 and r_{U_i} ensures that M_2 and M_3 are distinct for each session. Moreover, other exchanged messages $\langle M_7, M_8, T_2 \rangle$, $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ and $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ are also different for each session due to the use of timestamps and random nonces. In addition, our scheme allows to update old TID_i with a new TID_i^{new} for each session while the message $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ is sent to U_i by the GWN . After receiving the message, SP_i of the user U_i calculates $TID_i^{new} = M_{15} \oplus h(TID_i || M_1 || T_3 || T_4)$ and replaces TID_i with TID_i^{new} in its memory. Due to this, TID_i in the login request messages are distinct for different sessions. Thus, our scheme avoids traceability of U_i and SD_j by an attacker.

5.3.2 Anonymity

Prior to sending the login request $\langle TID_i, M_2, M_3, T_1 \rangle$ to the GWN , U_i hides its identity ID_i in $M_1 = A^* \oplus RPW_i^* = h(ID_i$

$\|K_{GWN-U_i}$, M_2 and M_3 . The GWN also hides the identities of U_i and SD_j as it computes $M_6 = h(ID_{SD_j} \| K_{GWN-SD_j})$, $M_7 = E_{M_6}[ID_i, ID_{GWN}, r_{U_i}, r_{GWN}, h(M_4)]$ and $M_8 = h(M_6 \| T_2 \| ID_i \| ID_{SD_j} \| ID_{GWN} \| r_{GWN})$ and $M_{14} = E_{M_4}[r_{U_i}, r_{GWN}, r_{SD_j}, ID_{SD_j}, ID_{GWN}, h(M_6)]$. SD_j also hides its own identity by computing $M_{10} = h(h(ID_{SD_j} \| K_{GWN-SD_j}) \| T_3) \oplus r_{SD_j}$. If an attacker intercepts all the messages during login and authentication phases, he/she is unable to identify ID_i and ID_{SD_j} as these are protected by symmetric encryption and one-way cryptographic hash function $h(\cdot)$. Therefore, the user and smart device anonymity are preserved in our scheme.

5.3.3 Privileged-Insider Attack

Suppose \mathcal{A} is a malicious insider user of the RA , who knows ID_i and RPW_i , which were sent to RA by U_i during his/her registration phase. Note that $RPW_i = h(PW_i \| \sigma_i \| a) \oplus r$. We assume that \mathcal{A} obtains the smart phone SP_i of U_i only after the user registration phase is finished. \mathcal{A} can then extract all the information $\{TID_i, A_i^*, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t\}$ stored in SP_i using the power analysis attacks [41]. Note that the user U_i already deleted the information A_i from its smart phone SP_i at the end of the user registration phase described in Section 4.2. Hence, without having A_i , it is computationally hard for \mathcal{A} to derive the secret r as $r = A_i^* \oplus A_i$. As a result, without r , \mathcal{A} can not derive $h(PW_i \| \sigma_i \| a) = RPW_i \oplus r$. Furthermore, without knowing a , it is computationally infeasible to derive the biometric key σ_i as $h(ID_i \| \sigma_i) = B_i \oplus a$. As a consequence, without having a , σ_i and K_{GWN-U_i} , it is also computationally hard for \mathcal{A} to guess correctly the password PW_i of U_i from $C_i = h(ID_i \| RPW_i \| \sigma_i) = h(ID_i \| (h(ID_i \| K_{GWN-U_i}) \oplus h(PW_i \| \sigma_i \| a)) \| \sigma_i)$. In summary, it is computationally hard for \mathcal{A} to guess and verify correctly PW_i and σ_i from RPW_i, A_i^*, B_i and C_i due to the collision resistant property of $h(\cdot)$. Therefore, our scheme is secure against the privileged-insider attack.

5.3.4 Stolen Smart Phone Attack

Suppose the smart phone SP_i of U_i is lost or stolen by an attacker \mathcal{A} . \mathcal{A} can then extract all information $\langle TID_i, A_i^*, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t \rangle$ stored in SP_i using the power analysis attacks [41]. Note that $B_i = h(ID_i \| \sigma_i) \oplus a$, $RPW_i = RPW_i \oplus r = h(PW_i \| \sigma_i \| a)$, $C_i = h(ID_i \| RPW_i \| \sigma_i)$ and $A_i^* = A_i \oplus r = h(ID_i \| K_{GWN-U_i}) \oplus RPW_i$. To correctly guess ID_i and PW_i from B_i and C_i respectively, \mathcal{A} needs to know both a and r . Again, to know a from B_i , \mathcal{A} needs both ID_i and PW_i . Thus, it is computationally infeasible for \mathcal{A} to correctly guess both ID_i and PW_i as ID_i and PW_i are protected by the one-way hash function $h(\cdot)$. Therefore, our scheme is secure against such an attack.

5.3.5 Session Key Security

The session key $SK_{ij} = h[ID_i \| ID_{SD_j} \| ID_{GWN} \| r_{U_i} \| r_{GWN} \| r_{SD_j} \| h(M_4) \| h(h(ID_{SD_j} \| K_{GWN-SD_j}))]$ is calculated by both U_i and SD_j . The message $\{M_{10}, M_{11}, M_{12}, T_3\}$ sent by SD_j to GWN contains session key SK_{ij} as $M_{11} = h(SK_{ij} \| T_3)$. Suppose an attacker \mathcal{A} intercepts this message and tries to compute the session key $SK'_{ij} = h[ID_i \| ID_{SD_j} \| ID_{GWN} \| r_{U_i} \| r'_{GWN} \| r'_{SD_j} \| h(M_4) \| h(h(ID_{SD_j} \| K_{GWN-SD_j}))]$ by generating the random nonces $r'_{U_i}, r'_{GWN}, r'_{SD_j}$ and timestamp T'_3 .

However, the computation of SK'_{ij} is not possible for \mathcal{A} because he/she does not know the various identities $ID_i, ID_{SD_j}, ID_{GWN}$, secret key K_{GWN-SD_j} , $M'_4 = h(ID_i \| K_{GWN-U_i})$. Without the knowledge of these parameters, and due to the collision resistance property of $h(\cdot)$, it is very difficult for \mathcal{A} to obtain SK'_{ij} . Therefore, our scheme preserves the session key security.

5.3.6 User Impersonation Attack

Suppose there is an adversary \mathcal{A} , who has the lost/stolen smart phone SP_i of a legal user U_i , and knows all the information stored in SP_i by the help of power analysis attacks [41]. Assume that \mathcal{A} intercepts U_i 's login request $\langle TID_i, M_2, M_3, T_1 \rangle$ and tries to create another valid login request, say $\langle TID_i, M'_2, M'_3, T'_1 \rangle$ on behalf of U_i , using the current timestamp T'_1 of his/her system. To compute M'_2, M'_1 is required to compute as $M'_1 = A^* \oplus RPW_i^* = h(ID_i \| K_{GWN-U_i})$. Suppose \mathcal{A} generates random nonce r'_{U_i} . To calculate $M'_2 = M'_1 \oplus r'_{U_i}$ and $M'_3 = h(M'_2 \| T'_1 \| ID_i \| TID_i \| r'_{U_i})$, \mathcal{A} needs ID_i and K_{GWN-U_i} , which are infeasible for him/her to obtain them. Due to the one-way hash function $h(\cdot)$, it is computationally infeasible for \mathcal{A} to create valid login request $\langle TID_i, M'_2, M'_3, T'_1 \rangle$ on behalf of U_i , even he/she knows the all information from the lost/stolen SP_i . So, it is clear that our scheme is secure against the user impersonation attack.

5.3.7 GWN Impersonation Attack

Suppose an adversary \mathcal{A} intercepts the messages $\langle M_7, M_8, T_2 \rangle$ and $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$, and attempts to create other valid messages, say $\langle M'_7, M'_8, T'_2 \rangle$ and $\langle M'_{14}, M'_{15}, M'_{16}, T'_3, T'_4 \rangle$ on behalf of the GWN , where $M_7 = E_{M_6}[ID_i, ID_{GWN}, r_{U_i}, r_{GWN}, h(M_4)]$, $M_6 = h(ID_{SD_j} \| K_{GWN-SD_j})$, $M_4 = h(ID_i \| K_{GWN-U_i})$ and $M_8 = h(M_6 \| T_2 \| ID_i \| ID_{SD_j} \| ID_{GWN} \| r_{GWN})$, $M_{14} = E_{M_4}[r_{U_i}, r_{GWN}, r_{SD_j}, ID_{SD_j}, ID_{GWN}, h(M_6)]$, $M_{15} = TID_i^{new} \oplus h(TID_i \| M_4 \| T_3 \| T_4)$, $M_{16} = h(M_{11} \| T_4 \| r_U)$. Suppose T'_2, T'_3, T'_4 and $r'_{U_i}, r'_{GWN}, r'_{SD_j}$ are the current timestamps and different random nonces generated by \mathcal{A} . To compute M'_7, M'_6, M'_4 and M'_8 , the secret key K_{GWN-SD_j} , and various identities ID_i, ID_{SD_j} and ID_{GWN} are required. To calculate M'_{14}, M'_{15} and M'_{16} , the secret key K_{GWN-U_i} , and various identities TID_i, ID_i, ID_{SD_j} and ID'_{GWN} are required. Moreover, the messages are protected by the one-way hash function $h(\cdot)$. Thus, \mathcal{A} is not able to create other valid messages $\langle M'_7, M'_8, T'_2 \rangle, \langle M'_{14}, M'_{15}, M'_{16}, T'_3, T'_4 \rangle$ on behalf of the GWN . Therefore, the proposed scheme is secure against the GWN impersonation attack.

5.3.8 Smart Device Impersonation Attack

Suppose an adversary \mathcal{A} intercepts the message $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ and attempts to create another valid message, say $\langle M'_{10}, M'_{11}, M'_{12}, T'_3 \rangle$ on behalf of the smart device SD_j , where T'_3 is the current timestamp of \mathcal{A} 's system when this message is created. Note that $M_{10} = h(h(ID_{SD_j} \| K_{GWN-SD_j}) \| T'_3) \oplus r'_{SD_j}$, $M_{11} = h(SK'_{ij} \| T'_3)$, $SK'_{ij} = h[ID_i \| ID_{SD_j} \| ID_{GWN} \| r'_{U_i} \| r'_{GWN} \| r'_{SD_j} \| h(M_4) \| h(h(ID_{SD_j} \| K_{GWN-SD_j}))]$, $M_{12} = h(r'_{SD_j} \| r'_{GWN} \| ID_{SD_j} \| ID_{GWN} \| T'_3)$ and $M'_4 = h(ID_i \| K_{GWN-U_i})$, where r'_{U_i}, r'_{GWN} and r'_{SD_j} are the random nonces created by \mathcal{A} . To calculate M'_{10}, M'_{11} and M'_{12} , the secret keys K_{GWN-SD_j} and $h(ID_i \| K_{GWN-U_i})$, and various identities ID_i, ID_{SD_j} and

ID_{GWN} are necessary. Therefore, \mathcal{A} is not able to create another valid message $\langle M'_{10}, M'_{11}, M'_{12}, T'_3 \rangle$ on behalf of SD_j . This confirms that the proposed scheme is secure against this attack.

5.3.9 Resilience against Smart Device Capture Attack

Suppose a smart device SD_j is physically captured by an attacker \mathcal{A} . Each SD_j contains the information $\{ID_{SD_j}, h(ID_{SD_j} || K_{GWN-SD_j})\}$. Since each K_{GWN-SD_j} is distinct, $h(ID_{SD_j} || K_{GWN-SD_j})$ is also distinct for each SD_j . If \mathcal{A} tries to extract K_{GWN-SD_j} from $h(ID_{SD_j} || K_{GWN-SD_j})$ using ID_{SD_j} , it is difficult task for \mathcal{A} to compute K_{GWN-SD_j} as K_{GWN-SD_j} is a long 1024-bit secret key. However, \mathcal{A} can know the session key SK_{ij} shared with the legal user U_i , which is stored in SD_j 's memory. Thus, compromise of this particular smart device SD_j in the smart home network does not lead to compromise of the session keys between that U_i and other non-compromised smart devices SD_l 's as the stored $h(ID_{SD_l} || K_{GWN-SD_l})$ is distinct for SD_l . The proposed scheme is then unconditionally secure against this attack.

5.3.10 Gateway Bypass Attack

In our scheme, both U_i and SD_j can not bypass the GWN due to the following argument. U_i can only send the login request through the GWN , and SD_j can send the authentication response only through the GWN . Both U_i and SD_j also establish the session key SK_{ij} through the GWN . When the GWN receives login request from U_i , it computes $M_7 = E_{M_6} [ID_i, ID_{GWN}, r_{U_i}^*, r_{GWN}, h(M_4)]$ and $M_8 = h(M_6 || T_2 || ID_i || ID_{SD_j} || ID_{GWN} || r_{GWN})$ and sends $\langle M_7, M_8, T_2 \rangle$ to SD_j , where $M_6 = h(ID_{SD_j} || K_{GWN-SD_j})$, and T_2 is the current timestamp generated by U_i . U_i can not compute M_6 as he/she does not know K_{GWN-SD_j} and it is only known to the GWN . Therefore, U_i is not able to compute M_7 and M_8 . When the GWN receives authentication reply from SD_j , it computes $M_{14} = E_{M_4} [r_{U_i}^*, r_{GWN}, r_{SD_j}^*, ID_{SD_j}, ID_{GWN}, h(M_6)]$, $M_{15} = TID_i^{new} \oplus h(TID_i || M_4 || T_3 || T_4)$, $M_{16} = h(M_{11} || T_4 || r_{U_i}^*)$ and sends the message $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ to U_i . SD_j can not compute M_4 as he/she does not know K_{GWN-U_i} . Therefore, SD_j can not compute M_{14} and M_{15} . To compute M_{16} , even if SD_j chooses current timestamp T_4 to compute $M_{16} = h(M_{11} || T_4 || r_{U_i}^*)$, but he/she does not know the random nonce $r_{U_i}^*$ of the user U_i . So, SD_j can not compute M_{14} , M_{15} and M_{16} . As a result, neither U_i nor GWN bypass the GWN in our proposed scheme.

5.3.11 Offline-Dictionary Attack

We consider an interesting attack scenario in our proposed scheme as illustrated by Huang et al. [34] to verify whether an adversary \mathcal{A} can derive the password of a legal user U_i or not. As in [34], we also consider the following attacking scenario as follows:

- At time T_1 , suppose U_i invokes the password and biometric update phase to change the password to PW_{i1} . At the end of this phase, the smart phone SP_i of U_i contains the information $\langle TID_i, A_i^*, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t \rangle$, where $A_i^* = h(ID_i || K_{GWN-U_i}) \oplus h(PW_{i1} || \sigma_{i1} || a)$ and σ_{i1} is the biometric key derived from the new biometrics BIO_{i1} entered by U_i at this time.

TABLE 2
Various Simulation Parameters

Parameter	Description
Platform	Ubuntu 14.04 LTS
Network coverage area	400 × 200 m ²
Network scenarios	1, 2 and 3
Number of users (U_i)	2,3,8 for scenarios 1,2,3
Number of gateway nodes (GWN)	1 for all scenarios
Number of smart devices (SD_j)	50 for all scenarios
Mobility	2 mps, 10 mps, 15 mps
Simulation time	1800 seconds
Routing protocol	AODV
Communication range of GWN	200 m
Communication range of SD_j	50 m

- At some time later (say, T_2), U_i again changes his/her password PW_1 to a new password PW_2 . At the end of this phase, the SP_i of U_i contains the information $\langle TID_i, A_i^{**}, B_i, C_i, \tau_i, h(\cdot), Gen(\cdot), Rep(\cdot), t \rangle$, where $A_i^{**} = h(ID_i || K_{GWN-U_i}) \oplus h(PW_{i2} || \sigma_{i2} || a)$ and σ_{i2} is the biometric key derived from the new biometrics BIO_{i2} entered by U_i at this time T_2 .
- A passive adversary \mathcal{A} with smart phone can obtain the data stored in the smart phone at time T_1 and T_2 .

Now, given (A_i^*, A_i^{**}) , \mathcal{A} can calculate $A_i^* \oplus A_i^{**} = h(PW_{i1} || \sigma_{i1} || a) \oplus h(PW_{i2} || \sigma_{i2} || a)$. By testing all password pairs in the password dictionary, \mathcal{A} can try to find at least one pair (pw_1, pw_2) such that $A_i^* \oplus A_i^{**} = h(pw_1 || \sigma_{i1} || a) \oplus h(pw_2 || \sigma_{i2} || a)$. However, to satisfy this condition, \mathcal{A} further needs to guess correctly the biometric keys pair $(\sigma_{i1}, \sigma_{i2})$. In addition, \mathcal{A} also needs the random secret a which is only known to U_i . To derive a , \mathcal{A} requires to guess the biometric key too. Thus, without having the biometric keys pair $(\sigma_{i1}, \sigma_{i2})$ and random secret a , it is computationally infeasible problem for \mathcal{A} to verify whether the guessed passwords pair (pw_1, pw_2) is correct or not. As a result, the proposed scheme has the ability to protect the offline-dictionary attack described in [34].

6 PRACTICAL PERSPECTIVE: NS2 SIMULATION

The proposed scheme is simulated using the widely-accepted networking simulation tool, NS2 2.35 simulator [16] on Ubuntu 14.04 LTS platform.

6.1 Simulation Parameters

The various simulation parameters are given in Table 2. The network coverage area is taken as 400 × 200 m². The communication ranges of the gateway node (GWN) and smart devices (SD_j) are taken as 200 m and 50 m, respectively. The network simulation time is taken as 1800 seconds (30 minutes). The traditional Ad hoc On-Demand Distance Vector (AODV) routing protocol is used as the routing protocol. Two types of users are taken in the simulation: first type consists of the static users, who do not move (for example, some smart home users seat on the chair and access SD_j), while the second type has moving users (for example, somebody is walking in the garden and accessing SD_j , or somebody is driving the card and accessing SD_j). The speeds for these smart home users are considered as 2, 10 and 15 mps, respectively.

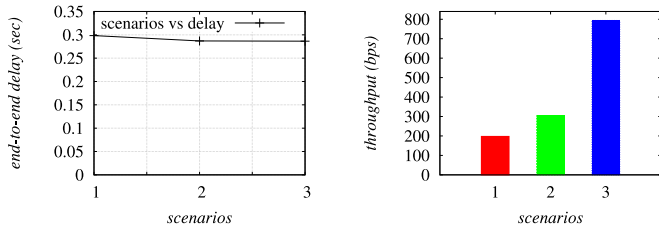


Fig. 8. (a) End-to-end delay (b) Throughput.

6.2 Simulation Environment

We have considered the following three network scenarios in the simulation. For all the scenarios, we have taken one GWN and 50 SD_j s.

Scenario 1. In this case, we have taken two users (U_i s): one is static and other one is moving with 2 mps.

Scenario 2. In this case, we have taken three users (U_i s): one is static and other two are moving with the speeds of 2 mps and 15 mps, respectively.

Scenario 3. In this case, we have taken eight users (U_i s): four are static and other four are moving with the speeds of 2 mps, 2 mps, 10 mps and 15 mps, respectively.

Moreover, we assume that the bit lengths of the identity, hash output (if we use SHA-1 hash algorithm) and random number/nonce are 128, 160 and 128 bits, respectively. In each scenario, we have considered the following messages between different network entities: $\langle TID_i, M_2, M_3, T_1 \rangle$, $\langle M_7, M_8, T_2 \rangle$, $\langle M_{10}, M_{11}, M_{12}, T_3 \rangle$ and $\langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ of sizes 480 bits, 960 bits, 512 bits and 1280 bits, respectively.

6.3 Simulation Results and Discussions

The network performance parameters, such as end-to-end delay (in seconds) and throughput (in bps) are calculated during the simulation.

6.3.1 Impact on End-to-End Delay

The end-to-end delay (EED) is calculated as the average time taken by the data packets to arrive at the destination from the source. The EED s of our scheme for different scenarios are given Fig. 8a. The EED s are 0.29832, 0.28687 and 0.28637 seconds for the network scenarios 1, 2 and 3, respectively. Note that the EED decreases in the scenarios 2 and 3, because in these scenarios we have considered more number of mobile users who are traveling towards the gateway node as compared to the scenario 1. For this reason, the EED reduces as the distance between the gateway node and mobile users decreases which affects the reducibility of the EED s accordingly.

6.3.2 Impact on Throughput

The throughput is measured as the number of bits transmitted per unit time. Fig. 8b depicts the network throughput (in bps) of our scheme under different network scenarios. The throughput values are 197.56, 303.87 and 793.78 bps for the scenarios 1, 2 and 3, respectively. Note that the throughput increases with an increase in the number of users. Due to the large number of users, more number of messages are exchanged in the network, and as a result, the throughput also increases.

TABLE 3
Communication Cost Comparisons

Scheme	Total messages	Total cost (bits)
Kumar et al. [5]	3	1696
Vaidya et al. [21]	2	2272
Kim-Kim [22]	2	4352
Jeong et al. [20]	2	1568
Santoso-Vun [25]	3	4416
Our	4	3232

7 PERFORMANCE COMPARISON

In this section, the proposed scheme is compared with related existing schemes of Kumar et al. [5], Vaidya et al. [21], Kim and Kim [22], Jeong et al. [20], and Santoso and Vun [25] during the login, and authentication and key agreement phases. Since the registration, and password and biometric update phases are not frequent, the costs involved in these phases are not discussed.

The communication costs of different existing schemes and our scheme are compared in Table 3. We have made a reasonable assumption that the identities are 128 bits in length; random nonces are 128 bits; timestamps are 32 bits; plaintext/ciphertext block in symmetric encryption/decryption (using AES-CBC algorithm) is 128 bits, and the hash digest is of 160 bits (if we use SHA-1 as $h(\cdot)$ [42]). By considering these values, the communication costs for the schemes of Kumar et al., Vaidya et al., Kim-Kim, Jeong et al., Santoso-Vun and our scheme are 1696, 2272, 4352, 1568, 4416, and 3232 bits, respectively. Note that in our scheme, the messages $MSG_1 = \langle TID_i, M_2, M_3, T_1 \rangle$, $MSG_2 = \langle M_7, M_8, T_2 \rangle$, $MSG_3 = \langle M_{10}, M_{11}, M_{12}, T_3 \rangle$, $MSG_4 = \langle M_{14}, M_{15}, M_{16}, T_3, T_4 \rangle$ are used. The cost of M_7 is $\lceil (128 + 128 + 128 + 128 + 160)/128 \rceil \times 128 = 768$ bits. Similarly, M_{14} needs $\lceil (128 + 128 + 128 + 128 + 128 + 160)/128 \rceil \times 128 = 896$ bits. So, the communication costs of different messages MSG_1 , MSG_2 , MSG_3 and MSG_4 are 480 bits, 960 bits, 512 bits, and 1280 bits, respectively. As a result, the total communication cost of the proposed scheme turns out to be $(480 + 960 + 512 + 1280) = 3232$ bits. Though our scheme requires more communication cost as compared to that for the schemes of Kumar et al., Vaidya et al. and Jeong et al., it is justified as our scheme supports additional functionality and security features (see Table 5).

In Table 4, we have used the notations T_{exp} , T_E/T_D , T_h , T_{fe} , T_{mac} and T_{hmac} to denote the computational time for modular exponentiation operation, symmetric encryption/decryption, hash function $h(\cdot)$ (using SHA-1 hashing

TABLE 4
Computation Costs Comparison

Scheme/phase	Total cost	Rough estimation
Kumar et al. [5]	$2T_h + T_{mac}$ $+ 1T_{hmac} + 2T_E/T_D$	12.48 ms
Vaidya et al. [21]	$20T_h + 3T_E/T_D$	23.20 ms
Kim-Kim [22]	$30T_h + 3T_E/T_D$	26.40 ms
Jeong et al. [20]	$10T_h + 3T_E/T_D$	20.00 ms
Santoso-Vun [25]	$2T_h + 3T_{exp}$	58.24 ms
Our	$22T_h + 4T_E/T_D + T_{fe}$	46.54 ms

TABLE 5
Security and Functionality Features Comparison

Functionality features	[5]	[21]	[22]	[20]	[25]	Our
SFF_1	✓	×	×	×	×	✓
SFF_2	×	×	×	×	✓	✓
SFF_3	×	✓	✓	✓	×	✓
SFF_4	✓	✓	✓	✓	✓	✓
SFF_5	×	×	×	×	×	✓
SFF_6	N/A	×	×	×	×	✓
SFF_7	N/A	✓	×	×	×	✓
SFF_8	✓	×	×	×	×	✓
SFF_9	N/A	×	×	×	×	✓
SFF_{10}	×	×	✓	✓	×	✓
SFF_{11}	✓	×	×	×	×	✓
SFF_{12}	✓	✓	✓	✓	✓	✓
SFF_{13}	✓	×	×	×	✓	✓
SFF_{14}	✓	✓	✓	✓	✓	✓
SFF_{15}	N/A	×	✓	×	×	✓
SFF_{16}	N/A	✓	×	×	×	✓
SFF_{17}	✓	✓	✓	✓	✓	✓
SFF_{18}	✓	✓	✓	✓	✓	✓
SFF_{19}	✓	✓	✓	✓	×	✓
SFF_{20}	✓	×	×	×	×	✓
SFF_{21}	✓	×	×	×	×	✓
SFF_{22}	N/A	✓	✓	✓	×	✓
SFF_{23}	N/A	×	×	×	×	✓
SFF_{24}	×	×	×	×	×	✓
SFF_{25}	✓	×	×	×	×	✓

Note: SFF_1 : mutual authentication between GWN and smart device; SFF_2 : mutual authentication between user and smart device; SFF_3 : mutual authentication between user and GWN; SFF_4 : key agreement; SFF_5 : traceability property; SFF_6 : password guessing attack; SFF_7 : password change attack; SFF_8 : dynamic smart device addition phase; SFF_9 : user anonymity property; SFF_{10} : GWN anonymity property; SFF_{11} : smart device anonymity property; SFF_{12} : replay attack; SFF_{13} : privileged-insider attack; SFF_{14} : man-in-the-middle attack; SFF_{15} : stolen smart phone/smart card attack; SFF_{16} : user impersonation attack; SFF_{17} : smart device impersonation attack; SFF_{18} : GWN bypassing attack; SFF_{19} : DoS attack; SFF_{20} : resilient against smart device capture attack; SFF_{21} : offline smart device registration phase; SFF_{22} : password change phase; SFF_{23} : biometric update phase; SFF_{24} : formal security proof under ROR model; SFF_{25} : formal security verification using AVISPA.

✓: the scheme is secure or supports a particular functionality/security feature; ×: the scheme is not secure or does not support a particular functionality/security feature. N/A: not applicable in the scheme.

algorithm), $Gen(\cdot)/Rep(\cdot)$, message authentication code (MAC) and hashed MAC, respectively. The bitwise XOR operation execution time is negligible, and we do not consider it as a performance evaluation parameter. The existing experimental values of these operations are given as follows in [43], [44]: T_{exp} , T_h , T_E/T_D , and T_{fe} are 0.0192 s, 0.00032 s, 0.0056 s and 0.0171 s, respectively. It is further assumed that $T_{mac} \approx T_{hmac} \approx T_h$. The computational costs of various schemes are given in Table 4. The total computational cost for our scheme is $22T_h + 4T_E/T_D + T_{fe}$, whereas the computational cost for a smart device is $7T_h + T_D \approx 7.84$ ms only. This indicates that our scheme is suitable for resource-constrained smart devices. The computation cost of our scheme is more than that for the schemes of Kumar et al., Vaidya et al., Kim-Kim and Jeong et al., because we have used the fuzzy extractor for providing additional security level of the system as compared to other schemes. However, our scheme provides extra functionality features and security features, and the cost for a resource constrained smart device is low.

Finally, the functionality and security features comparison among our scheme and other schemes is shown in Table 5. The scheme of Vaidya et al. is insecure against privileged-insider, password guessing, and smart device capture attacks, and it does not have the traceability, user anonymity and smart device anonymity properties. Moreover, the dynamic smart device addition phase, offline smart device registration phase, formal security proof under standard model and formal security verification using AVISPA are not supported in their scheme. Kim-Kim's scheme is vulnerable to password guessing attack, password change attack, privileged-insider attack, user impersonation attack through privileged-insider attack and smart device capture attack, and it does not have traceability, user anonymity and smart device anonymity properties. Additionally, the dynamic smart device addition phase, offline smart device registration phase, formal security proof under the ROR model and formal security verification using AVISPA are not available in Kim-Kim's scheme. Kumar et al. does not support traceability and gateway anonymity properties and it does not provide formal security proof under the ROR model. The schemes of Kumar et al., Jeong et al. and Santoso-Vun also lack the functionality features, which are shown in Table 5. In summary, our scheme provides significantly better security and functionality features as compared to those for other existing schemes.

8 CONCLUSION

This paper presents a new scheme to address the user authentication issue in a smart home environment. The proposed scheme provides additional functionality features. The proposed scheme is secure against several known attacks, which are shown through random oracle model, informal security and AVISPA tool. The practical implementation of the proposed scheme is also demonstrated though the widely-accepted NS-2 simulator. Overall, the proposed scheme provides a better trade-off between security and functionality features provided in Table 5, and overheads as compared to other existing related schemes.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and the Associate Editor for providing constructive and generous feedback. This work was supported by the Information Security Education & Awareness (ISEA) Phase II Project, Department of Electronics and Information Technology (DeitY), India. The work done in this paper is also supported from the research grant from CSIR, New Delhi with letter number: 22/717/16/EMR-II.

REFERENCES

- [1] What is a Smart Home? [Online]. Available: <http://smarthomeenergy.co.uk/what-smart-home>, Accessed on: Apr. 2016.
- [2] C. Gomez and J. Paradells, "Wireless home automation networks: A survey of architectures and technologies," *IEEE Commun. Mag.*, vol. 48, no. 6, pp. 92–101, Jun. 2010.
- [3] J. E. Kim, G. Boulous, J. Yackovich, T. Barth, C. Beckel, and D. Mosse, "Seamless integration of heterogeneous devices and access control in smart homes," in *Proc. 8th Int. Conf. Intell. Environ.*, 2012, pp. 206–213.

- [4] A. Lazakidou, *Wireless Technologies for Ambient Assisted Living and Healthcare: Systems and Applications: Systems and Applications*. Hershey, PA, USA: IGI Global, 2010.
- [5] P. Kumar, A. Gurtov, J. Iinatti, M. Ylianttila, and M. Sain, "Lightweight and Secure session-key establishment scheme in smart home environments," *IEEE Sensors J.*, vol. 16, no. 1, pp. 254–264, Jan. 2016.
- [6] R. Volner, P. Bore, and V. Smrz, "A product based security model for smart home appliances," in *Proc. 11th Int. Biennial Baltic Electron. Conf.*, 2008, pp. 221–222.
- [7] H. Tschofenig, J. Arkko, and D. McPherson, "Architectural considerations in smart object networking, Internet Engineering Task Force, RFC-7452," Internet Eng. Task Force, Fremont, CA, USA, 2014.
- [8] I. Bierhoff, et al., "Towards an inclusive future—Impact and wider potential of information and communication technologies," in *Smart Home Environment*. Brussels, Belgium: East Sussex Press, 2007.
- [9] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inform. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.
- [10] E. Bertino, N. Shang, and S. S. W. Jr, "An efficient time-bound hierarchical key management scheme for secure broadcasting," *IEEE Trans. Dependable Secure Comput.*, vol. 5, no. 2, pp. 65–70, Apr. 2008.
- [11] S. Frankel, R. Glenn, and S. Kelly, "The AES-CBC cipher algorithm and its use with IPsec," 2013. [Online]. Available: <http://tools.ietf.org/html/rfc3602>, Accessed on: Jul. 2017.
- [12] Advanced Encryption Standard (AES), FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001. [Online]. Available: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>, Accessed on: Apr. 2016.
- [13] R. Canetti, "Introduction to Cryptography. Lecture 9 - Symmetric Encryption," 2008. [Online]. Available: <http://www.cs.tau.ac.il/canetti/f08-materials/scribe9.pdf>, Accessed on: Jul. 2017.
- [14] M. Abdalla, P. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proc. 8th Int. Workshop Theory Practice Public Key Cryptography*, 2005, pp. 65–84.
- [15] AVISPA, "Automated Validation of Internet Security Protocols and Applications," [Online]. Available: <http://www.avispa-project.org/>, Accessed on: Apr. 2016.
- [16] The Network Simulator-ns-2. [Online]. Available: <http://www.isi.edu/nsnam/ns/>, Accessed on: Apr. 2016.
- [17] P. Sarkar, "A simple and generic construction of authenticated encryption with associated data," *ACM Trans. Inform. Syst. Secur.*, vol. 13, no. 4, pp. 1–16, 2010, Art. No. 33.
- [18] K.-K. R. Choo, "Key establishment: Proofs and refutations," Ph.D. dissertation, Queensland University of Technology, Brisbane, Australia, 2006. [Online]. Available: http://eprints.qut.edu.au/16262/1/Kim-Kwang_Choos_Thesis.pdf
- [19] S. Wu and K. Chen, "An efficient key-management scheme for hierarchical access control in E-medicine system," *J. Med. Syst.*, vol. 36, no. 4, pp. 2325–2337, 2012.
- [20] J. Jeong, M. Y. Chung, and H. Choo, "Integrated OTP-based user authentication scheme using smart cards in home networks," in *Proc. 41st Annu. Hawaii Int. Conf. Syst. Sci.*, 2008, pp. 294–294.
- [21] B. Vaidya, J. H. Park, S. S. Yeo, and J. J. Rodrigues, "Robust one-time password authentication scheme using smart card for home network environment," *Comput. Commun.*, vol. 34, no. 3, pp. 326–336, 2011.
- [22] H. J. Kim and H. S. Kim, "AUTH_HOTP - HOTP based authentication scheme over home network environment," in *Proc. Int. Conf. Comput. Sci. Appl.*, 2011, pp. 622–637.
- [23] B. Vaidya, D. Makrakis, and H. T. Mouftah, "Device authentication mechanism for smart energy home area networks," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2011, pp. 787–788.
- [24] P. Hanumanthappa and S. Singh, "Privacy preserving and ownership authentication in ubiquitous computing devices using secure three way authentication," in *Proc. Int. Conf. Innovations Inform. Technol.*, 2012, pp. 107–112.
- [25] F. K. Santos and N. C. H. Vun, "Securing IoT for smart home system," in *Proc. Int. Symp. Consum. Electron.*, 2015, pp. 1–2.
- [26] C.-C. Chang and H.-D. Le, "A provably secure, efficient and flexible authentication scheme for ad hoc wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 1, pp. 357–366, Jan. 2016.
- [27] A. K. Das, S. Kumari, V. Odelu, X. Li, F. Wu, and X. Huang, "Provably secure user authentication and key agreement scheme for wireless sensor networks," *Secur. Commun. Netw.*, vol. 9, no. 16, pp. 3670–3687, 2016.
- [28] Y. Li, "Design of a key establishment protocol for smart home energy management system," in *Proc. 5th Int. Conf. Comput. Intell. Commun. Syst. Netw.*, 2013, pp. 88–93.
- [29] K. Han, J. Kim, T. Shon, and D. Ko, "A novel secure key paring protocol for RF4CE ubiquitous smart home systems," *Pers. Ubiquitous Comput.*, vol. 17, no. 5, pp. 945–949, 2012.
- [30] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn. Advances Cryptology*, 2004, pp. 523–540.
- [31] V. Odelu, A. K. Das, and A. Goswami, "A secure biometrics-based multi-server authentication protocol using smart cards," *IEEE Trans. Inform. Forensics Secur.*, vol. 10, no. 9, pp. 1953–1966, Sep. 2015.
- [32] A. K. Das, "A secure user anonymity-preserving three-factor remote user authentication scheme for the telecare medicine information systems," *J. Med. Syst.*, vol. 39, no. 3, 2015, Art. no. 30.
- [33] A. K. Das, "A secure and robust temporal credential-based three-factor user authentication scheme for wireless sensor networks," *Peer-to-Peer Netw. Appl.*, vol. 9, no. 1, pp. 223–244, 2016.
- [34] X. Huang, X. Chen, J. Li, Y. Xiang, and L. Xu, "Further observations on smart-card-based password-authenticated key agreement in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1767–1775, Jul. 2014.
- [35] S. Chatterjee, S. Roy, A. K. Das, S. Chattopadhyay, N. Kumar, and A. V. Vasilakos, "Secure Biometric-Based Authentication Scheme using Chebyshev Chaotic Map for Multi-Server Environment," *IEEE Trans. Dependable and Secure Comput.*, 2016, DOI: 10.1109/TDSC.2016.2616876.
- [36] A. K. Das, "A secure and effective biometric-based user authentication scheme for wireless sensor networks using smart card and fuzzy extractor," *Int. J. Commun. Syst.*, vol. 30, no. 1, pp. 1–25, 2017.
- [37] A. Armando, et al., "The AVISPA tool for the automated validation of internet security protocols and applications," in *Proc. 17th Int. Conf. Comput. Aided Verification*, 2005, pp. 281–285.
- [38] AVISPA, "SPAN, the Security Protocol ANimator for AVISPA." [Online]. Available: <http://www.avispa-project.org/>, Accessed on: Oct. 2016.
- [39] D. von Oheimb, "The high-level protocol specification language HLPSP developed in the EU project AVISPA," in *Proc. 3rd APPSEM II (Appl. Semantics II) Workshop*, 2005, pp. 1–17.
- [40] V. Odelu, A. K. Das, and A. Goswami, "SEAP: Secure and efficient authentication protocol for NFC applications using pseudonyms," *IEEE Trans. Consum. Electron.*, vol. 62, no. 1, pp. 30–38, Feb. 2016.
- [41] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.
- [42] Secure Hash Standard, FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, April 1995. [Online]. Available: <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>, Accessed on: Sep. 2015.
- [43] D. He, N. Kumar, J. H. Lee, and R. S. Sherratt, "Enhanced three-factor security protocol for consumer USB mass storage devices," *IEEE Trans. Consum. Electronics*, vol. 60, no. 1, pp. 30–37, Feb. 2014.
- [44] C.-C. Lee, C. T. Chen, P. H. Wu, and T. Y. Chen, "Three-factor control protocol based on elliptic curve Cryptosystem for universal serial bus mass storage devices," *IET Comput. Digital Techn.*, vol. 7, pp. 48–55, 2013.



Mohammad Wazid (S'17) received the MTech degree in computer network engineering from Graphic Era University, Dehradun, India and the PhD degree in computer science and engineering from the International Institute of Information Technology (IIIT), Hyderabad, India. His current research interests include security in wireless sensor network, vehicular adhoc network, Internet of Things (IoT) and cloud computing. He has published more than 50 papers in international journals and conferences in the above areas. He is a member of the IEEE.



Ashok Kumar Das (M'17) received the MSc degree in mathematics, the MTech degree in computer science and data processing, and the PhD degree in computer science and engineering from IIT Kharagpur, India. He is currently an assistant professor with the Center for Security, Theory and Algorithmic Research, IIIT, Hyderabad, India. His current research interests include security in wireless sensor network, vehicular ad hoc networks, smart grid, Internet of Things (IoT) and cloud computing. He has authored more than 145 papers in

international journals and conferences in the above areas. He was a recipient of the Institute Silver Medal from IIT Kharagpur. He is in the editorial board of the *KSII Transactions on Internet and Information Systems*, and the *International Journal of Internet Technology and Secured Transactions* (Inderscience). He is a member of the IEEE.



Vanga Odelu received the MTech degree in computer science and data processing and PhD degree from IIT Kharagpur, India. He is currently an assistant professor in the Department of Computer Science and Engineering, Indian Institute of Information Technology, Sri City, India. His research interests include user authentication, security in cloud computing and smart grid. He has authored more than 40 papers in international journals and conferences.



Neeraj Kumar (M'16, SM'17) received the PhD degree in computer science engineering from Shri Mata Vaishno Devi University, Katra, India, in 2009. He is currently an Associate Professor in the Department of Computer Science and Engineering, Thapar University, Patiala, India. He has guided many students leading to ME and PhD. He has more than 200 technical research papers in leading journals such as the *IEEE Transactions on Industrial Informatics*, the *IEEE Transactions on Industrial Electronics*, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Intelligent Transportation Systems*, the *IEEE Transactions on Power Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Transaction on Information Forensics and Security*, *IEEE Transactions on Smart Grid*, the *IEEE SYSTEMS JOURNAL*, the *IEEE Communications Magazine*, the *IEEE Wireless Communications Magazine*, and conferences including IEEE ICC, IEEE Globecom etc. His research is supported by Department of Science and Technology, Tata Consultancy Services, and University Grants Commission. His research interests include mobile computing, parallel/distributed computing, multi-agent systems, service oriented computing, routing, and security issues in mobile ad hoc, sensor, and mesh networks. He is associate editor of JNCA, Elsevier, IJCS, Wiley and Security and Privacy, Wiley.

His research is supported by Department of Science and Technology, Tata Consultancy Services, and University Grants Commission. His research interests include mobile computing, parallel/distributed computing, multi-agent systems, service oriented computing, routing, and security issues in mobile ad hoc, sensor, and mesh networks. He is associate editor of JNCA, Elsevier, IJCS, Wiley and Security and Privacy, Wiley.



Willy Susilo (SM'02) received the PhD degree in computer science from the University of Wollongong, Australia. He is currently a professor and the head of the School of Computing and Information Technology with the University of Wollongong, Australia. He is also the director of the Centre for Computer and Information Security Research with the University of Wollongong. He has been awarded the Prestigious ARC Future fellow by the Australian Research Council. His main research interests include cloud security, cryptography, and

information security. He has served as a program committee member in major international conferences, including Asiacrypt and CT-RSA. He is the editor-in-chief of the *Information* journal. He is also an associate editor of the *IEEE Transactions on Information Forensics and Security*, the *Computer Standards & Interfaces*, and the *International Journal of Information Security*. He is a senior member of the IEEE.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

senDroid: Auditing Sensor Access in Android System-Wide

Weili Han¹, Member, IEEE, Chang Cao¹, Hao Chen, Dong Li¹, Zheran Fang, Wenyan Xu, Member, IEEE, and X. Sean Wang, Senior Member, IEEE

Abstract—Sensors are widely used in modern mobile devices (e.g., smartphones, watches) and may gather abundant information from environments as well as about users, e.g., photos, sounds and locations. The rich set of sensor data enables various applications (e.g., health monitoring) and personalized apps as well. However, the powerful sensing abilities provide opportunities for attackers to steal both personal sensitive data and commercial secrets like never before. Unfortunately, the current design of smart devices only provides a coarse access control on sensors and does not have the capability to audit sensing. We argue that knowing how often the sensors are accessed and how much sensor data are collected is the first-line defense against sensor data breach. Such an ability is yet to be designed. In this paper, we propose a framework that allows users to acquire sensor data usages. In particular, we leverage a hook-based track method to track sensor accesses. Thus, with no need to change the source codes of the Android system and applications, we can intercept sensing operations to graphic sensors, audio sensors, location sensors, and standard sensors, and audit them from four aspects: flow audit, frequency audit, duration audit and invoker audit. Then, we implement a prototype, referred to as *senDroid*, which visually shows the quantitative usages of these sensors in real time at a performance overhead of [0.04–8.05] percent. *senDroid* allows Android users to audit the applications even when they bypass the Android framework via JNI invocations or when the malicious codes are dynamically loaded from the server side. Our empirical study on 1,489 popular apps in three well-known Android app markets shows that 26.32 percent apps access sensors when the apps are launched, and 11.01 percent apps access sensors while the apps run in the background. Furthermore, we analyze the relevance between sensor usage patterns and third-party libraries, and reverse-engineering on suspicious third-party libraries shows that 77.27 percent apps access sensors via third-party libraries. Our results call attentions to address the users' privacy concerns caused by sensor access.

Index Terms—Android security, sensor, audit, hooking, senDroid

1 INTRODUCTION

WITH the development of sensing technologies, smartphones are increasingly equipped with sensors, such as cameras, microphones, GPS, motion sensors, etc. [1]. Already, a modern smartphone has more than ten high-accuracy sensors, and these sensors enable novel and exciting applications.

With the trend of environment-aware and user-oriented apps, mobile applications tend to gather an increasing amount of sensor data, and the users are facing a higher risk than never before. At the beginning of 2016, it is reported that *Alipay* silently took photos without informing its users [2], although *Alipay* removed this function after users become aware of it and showed their serious concerns. Furthermore, many sensor data allow applications to infer security-sensitive information, e.g., inferring keystrokes [3] and voice [4] from motion sensors.

Smartphones can audit the network traffic, yet they are incapable of measuring sensor accesses for identifying malicious or suspicious usages of sensors. An application could maliciously gather sensor data without users' consents, especially from the standard sensors (e.g., motion sensors), because they are not under the control of the Android permission mechanism. To investigate the legitimacy of an application's access to sensor data, it is necessary to monitor sensor accesses and raise an alert to the users. If inappropriate usage is found, for instance, sensor access monitoring can help to detect whether an application accesses a sensor when a user is not expecting it (e.g., secretly taking a photo), or whether it gathers an abnormally large amount of sensor data.

To the best of our knowledge, we are not aware of any framework that can comprehensively monitor the sensor data access pattern of applications, e.g., measuring when an application accesses sensors and how many data it has acquired. Although *taintDroid* [5] can detect which application is accessing a sensor, it does not measure how many data the application is reading. Measuring the sensor data access traffic is challenging, because it not only needs to be efficient and imposes small overhead but also is able to cope with circumvention by applications.

To audit the sensor access efficiently and effectively, we propose a framework, referred to as *senDroid*, to quantitatively measure sensor usages in Android. *senDroid* uses low-level

- W. Han, C. Cao, D. Li, Z. Fang, and X. S. Wang are with the Software School, Fudan University and the Shanghai Key Laboratory of Data Science, Shanghai 201203, China. E-mail: {wlhan, 16212010001, 14212010008, 13212010002, xywangCS}@fudan.edu.cn.
- H. Chen is with the Department of Computer Science University of California, Davis, CA 95616-5270. E-mail: chen@ucdavis.edu.
- W. Xu is with the Department of Electronic Engineering, Zhejiang University, Hangzhou, Zhejiang 310000, China. E-mail: xuwenyuan@zju.edu.cn.

Manuscript received 6 Nov. 2016; revised 12 Oct. 2017; accepted 13 Oct. 2017. Date of publication 1 Nov. 2017; date of current version 18 Mar. 2020.

(Corresponding author: Weili Han.)

Digital Object Identifier no. 10.1109/TDSC.2017.2768536

hooks to intercept all sensor-related API calls so that it measures when an application reads a sensor and how many data it reads. In summary, our contributions are listed as follows:

- We design a framework, referred to as *senDroid*, which can audit all four categories of sensors: graphics, audio, location, and standard sensors, in the Android platform. *senDroid* can intercept all accesses to sensors, including access via JNI (Java Native Interface). Our performance evaluation shows that *senDroid* incurs a moderate overhead of [0.04–8.05] percent on Android smartphones. Moreover, because *senDroid* hooks into the processes of both applications and Android system to intercept sensor-related API calls, it requires neither modification to the original Android systems nor the source code of the applications. Thus, *senDroid* can be widely deployed and is capable of detecting potential attacks that bypass the Android framework.
- Our experiments show that *senDroid* can report all sensor accesses by any applications, even by the dynamically loaded codes. We evaluate *senDroid* over 1,489 popular applications in three well-known Android application markets (two popular Chinese App markets and *Google Play Store*) and study *senDroid* in two running phases of applications—*Launch Phase* and *Silent Phase* (i.e., in the background). We observe the following:
 - During the *Launch Phase*, 25.35, 11.92, 0.46, and 0.46 percent applications in Chinese Android app markets access location sensors, standard sensors, graphic sensors, and audio sensors, respectively. These numbers are 12.70, 4.24, 0.46, and 0.46 percent higher than those in *Google Play Store*, respectively. When running in *Silent Phase*, 13.19 and 2.66 percent applications in Chinese Android app markets access location sensors and standard sensors, respectively, which is 8.55 and 1.54 percent higher than those in *Google Play Store*. These results show that applications in Chinese Android app markets access sensors more than those in *Google Play Store*, especially during the *Silent Phase*.
 - We find that third-party library is one of the main causes that leads to sensor access in *Silent Phase*. We reverse-engineer the applications that access sensors during *Silent Phase*, and find that 77.27 percent applications access sensors from third-party libraries, but such accesses caused by third-party libraries rarely appear in the apps' descriptions. We recommend that developers should disclose third-party libraries and their sensor access in the app's description.

The rest of paper is organized as follows. Section 2 introduces the background knowledge and the motivated scenarios of our paper; Section 3 presents the design of *senDroid*; Section 4 shows the details of *senDroid* implementation; Section 5 evaluates *senDroid* from accuracy and performance overheads; Section 6 empirically studies the popular applications in *Wandoujia*, *360* and *Google Play Store* to show the usages of sensors in reality; Section 7 discusses

the potential issues; Section 8 investigates the related research works; Section 9 summarizes the paper and introduces our future work.

2 BACKGROUND AND MOTIVATION

2.1 Sensors in the Android Platform

Each Android enabled devices, including smartphones, may contain more than ten sensors, such as cameras, microphones. The sensors in an Android-enabled device could be divided into four categories.

- *Graphic Sensors*. Graphic sensors refer to those sensors that collect graphic data, e.g., photos, videos. Android applications connect with graphic sensors via *CameraService* and achieve related operations, e.g., starting preview, taking a picture, or recording videos via *Camera* and *MediaRecorder*. In the Android permission system, a developer can request the permission of *CAMERA* to legally access graphic sensors.
- *Audio Sensors*. Audio sensors, e.g., microphone in the Android platform help devices to capture ambient sounds. In Android, there exists two ways to access the audio sensors. One is *AudioRecord* that provides raw sound streams to applications; and the other is *MediaRecorder* that offers compressed audio files. In the Android permission system, a developer can request the permission of *RECORD_AUDIO* to legally access audio sensors.
- *Location Sensors*. Location sensors primarily refer to the GPS, which collects the location data of a smartphone. Android applications utilize *LocationManager* to request the latest location or to get location periodically by registering *LocationListener*. In the Android permission system, a developer can request the permission of *ACCESS_FINE_LOCATION* or *ACCESS_COARSE_LOCATION* to legally access location sensors.
- *Standard Sensors*. The Android platform officially defines the standard sensors. Some (e.g., accelerometer and gyroscope) can monitor the motion of phones' users. Others can monitor various environmental properties, e.g., humidity, illuminance, pressure, temperature, and geomagnetic field. *SensorService* is the primary service of standard sensors. Android applications obtain the list of standard sensors that the platform supports, create connection (*SensorEventConnection*) and poll the standard sensors devices through *SensorService*.

Because the sensor data usually contains rich information, the vendors of mobile applications tend to gather more and more sensor data than never before. Many literatures [6], [7], [8], [9] focus on detecting the user location via standard sensors' data instead of GPS, and on revealing sound information with standard sensors data [4], [10]. Moreover, the standard sensors' data can be utilized for inferring user's inputs [3], [11], [12], and for identifying a device [13], [14] or even an individual [15].

Android permission system utilizes permissions to control accesses to sensors. However, many standard sensors (e.g., accelerometer and gyroscope), are not under control.

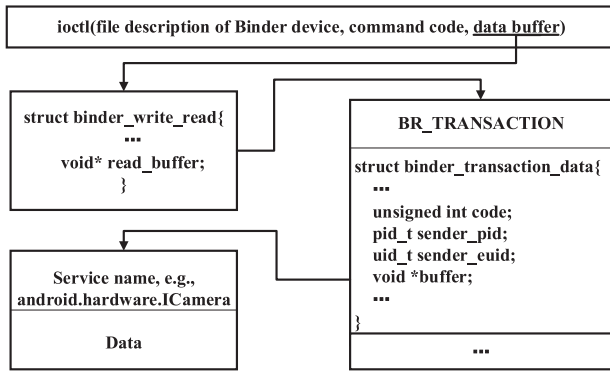


Fig. 1. Brief structures in `ioctl` called by Binder.

To the best of our knowledge, it is yet to carry out the empirical study to effectively evaluate the sensor usages of mainstream application markets.

2.2 Communication Between Applications and Sensors

2.2.1 Binder

Binder [16], originally named OpenBinder, is a system-level architecture for IPC (Inter-Process Communication). Binder inter-process communication framework follows a client-server architecture, and consists of four components: client, server, service manager, and Binder driver. Binder driver communicates with a Binder device by using the system call `ioctl`. Fig. 1 indicates the related data structures and API's arguments in `ioctl`, which consists of three parts: the file description of a Binder device, a Binder driver command code, and a data buffer. The major driver command code is `BINDER_WRITE_READ`. A Binder driver uses this command to read data from or write data to a Binder device. The data read or to be written is organized in a structure named `binder_write_read`, which comprises a series of pairs of a target command and an argument. The target command we concerned is `BR_TRANSACTION` which is shown in Fig. 1. This command is sent by the client, and its corresponding argument is a request whose structure is named `binder_transaction_data`. The variable code in the structure of `binder_transaction_data` is the command code negotiated by the sender and receiver. In general, the serial number of public interface is defined in the service. The uid of an Android application that initiates the transaction is shown in the variable `sender_euid`, which can help us to identify which Android application requests the sensor data. The extra arguments, including the name of service which will deal with the request, are stored in the field `buffer`.

2.2.2 Interaction with Sensor Devices

As described in Section 2.1, there can be more than ten sensors in an Android enable device. After an application sends a request to a corresponding service, the service will interact with the corresponding sensor by calling the standard interface defined in the HAL (Hardware Abstraction Layer), which provides a standard interface for hardware vendors to implement [17]. Each accessory, e.g., sensor, in Android is treated as a file so that it can be accessed using standard I/O system calls. The sensor driver that implements the

HAL interface opens the sensor devices and invokes `read`, `write` or `ioctl` system call to interact with the sensor devices. Android utilizes V4L2 (Video for Linux 2) as its camera driver, ALSA (Advanced Linux Sound Architecture) as its audio driver while no official GPS driver or standard sensors driver.

2.3 Hooking

Hooking is a technique of inserting codes into a system call for alteration. The typical hook works by replacing the function pointer to the call(s) a developer wants to intercept with that. Once it is done, it will then call the original function pointer [18]. There are two steps, function substitution and dynamic-link library injection, when we want to hook a system call.

2.3.1 Function Substitution

In Android, share libraries are ELF (Executable and Linkable Format) [19] files which are mapped into the memory space of a process at runtime. The GOT (Global Offset Table) and the PLT (Procedure Linkage Table) are two pivotal parts of ELF file. A GOT is a table of addresses in the data section. When an instruction in the code section refers a variable, it looks up the entries in the GOT, which keeps the absolute addresses of variables. Each entry of the PLT is a chunk of instructions corresponding to an external function the shared library calls [20]. When an instruction calls an external function, it calls an entry in the PLT, which then calls the actual function. The address of the actual function is determined by the corresponding entry in the GOT.¹ The instructions to call external function are essentially jump instructions pointing to entries in the PLT. Then, the PLT entries retrieve the absolute addresses of the functions, which are contained by the GOT entries. Due to the indirection to function references, the hooking can be achieved by substituting designate function pointers with the original ones in the GOT for each ELF file that the target process loaded.

2.3.2 Dynamic-Link Library Injection

Dynamic-link library (DLL) injection is a technique that allows a process to run codes in the address space of another process by forcing it to load a dynamic-link library [21]. In Android, an approach to realize DLL injection is to leverage the system call `ptrace`, which provides a process with the capability of monitoring and controlling the execution of another process [22].

2.4 Motivation Scenarios

Alice bought an Android phone with several applications pre-installed. She might also want to install several applications herself. The recent news reported that applications could steal information from on-board sensors in mobile devices. Worried that the applications on her phone could be malicious, she installed our *senDroid* on her phone. After running *senDroid* for several days, she can check the reports

1. The address contained by the corresponding GOT entry points to the PLT entry itself when first calling the function. It points to the actual function only when the dynamic loader resolves it. This mechanism is called lazy binding or lazy linking [20] while it is not adopted by the current version of Android.

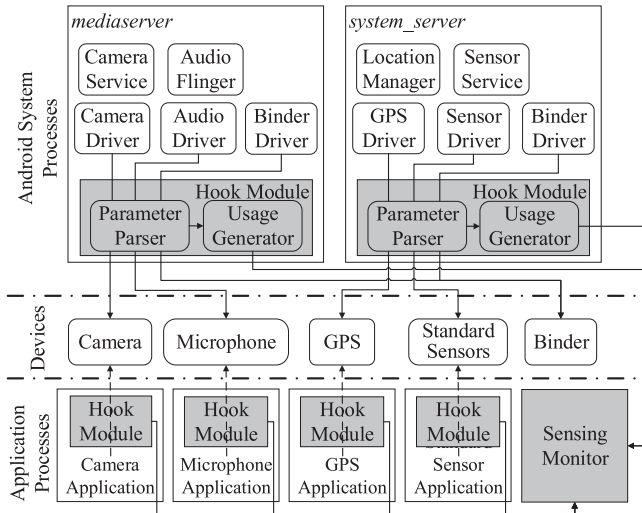


Fig. 2. Design of *senDroid* framework. The shadowed blocks are designed in *senDroid*.

of *senDroid*, and judge whether the installed applications stole sensors data. *senDroid* can help users to monitor and analyze the usages of sensors on a mobile device, with only negligible performance overhead but without the modification of Android framework.

In another scenario, Bob, who is a bouncer of an application market of Android, can use *senDroid* to detect suspicious applications in the market. After installing *senDroid*, he could install the suspicious applications on an Android phone, and operate the applications with the help of test tools. Then he may compare the log with the applications' description either manually or automatically with the supports of natural language processing.

3 SENDROID: DESIGN

3.1 Threat Model

The security threats considered in our work come from the third-party applications which access sensors data surreptitiously. These applications can be divided into two types:

- Malwares, e.g., applications implementing the potential attacks proposed in prior work [4], [23], [24], [25]. Some of them infer personal privacy from the unrestrained sensors (e.g., accelerometer and gyroscope), while the others access restrained sensors by an undetectable approach, e.g., taking photos or recording video by JNI without calling any Android API. Even some applications dynamically load remote codes to implement suspicious functions bypass applications' bouncers.
- Applications [26] using ad/analytics libraries that collect sensor data for precision advertising and improving user experience. But they do not declare their accesses in their description.

Note that, both types of applications can be suspicious but not vicious, which means that they would not require a highly-escalated privilege or circumvent the Android system (e.g., modifying or bypassing the SELinux policies on the device). Targeted at these honest but curious applications, and motivated to mitigate the above threats, we

TABLE 1
Audit capacities of *senDroid* for Four Categories Sensors

Sensor Type	Sensor Data	Flow Audit	Frequency Audit	Duration Audit	Invoker Audit
Graphic Sensors	Preview	✓	✓	✓	✓
	Video	✓	✓	✓	✓
	Photo	-	✓	-	✓
Audio Sensors	Audio	✓	✓	✓	✓
Location Sensors	Location	✓	✓	✓	✓
Standard Sensors	Motion and Environment	-	✓	✓	✓

propose *senDroid* to counter these suspicious behaviors by auditing the accesses to sensors on an Android device.

3.2 Framework Overview

Fig. 2 shows an overview of sensing audit in *senDroid*. Basically, *senDroid* consists of (1) *Hook Module*, a sentry deployed in system or application processes that intercepts the sensor-related traffic to reveal sensing usage; and (2) *Sensing Monitor*, an Android application that visualizes the sensing usages. The solid arrows in Fig. 2 represent the standard access mode of sensors. In the standard access mode, sensors applications send requests to corresponding services, and the services handle the sensors' data accesses. The dashed arrows represent a deviant access mode in which sensors applications directly access the sensor device or call the interface defined in HAL in virtue of JNI. To handle both standard and deviant access modes, the *Hook Module* is embedded in both Android system processes that contain sensors' service and application processes that request sensor data. Overall, *senDroid* interposes in the sensor data flow to perform four types of audit.

- *Flow Audit*. Every time accessing a sensor, an application gains the sensor data in various size. We monitor the size of sensor data to report the amount of sensor traffics the application requested the sensor.
- *Frequency Audit*. When accessing sensor data, an application may send requests with different frequencies. *senDroid* can reveal how often and what time the application accessed the sensor data.
- *Duration Audit*. In the continuous sensing case, we record the duration to profile the behavior of applications more comprehensively.
- *Invoker Audit*. *senDroid* audits which application initiates the sensor data access originally.

The audit capacities of *senDroid* for different categories of sensors are shown in Table 1. We will discuss the reasons why *senDroid* does not implement the flow audit of photo and standard sensors in Section 4.

3.3 Hook Module

A *Hook Module* consists of two main components: a *Parameter Parser* parses the data in intercepted system calls and a *Usage Generator* generates the usage reports of sensors according to the accessing time, data size and sensor accessor. Table 2 illustrates APIs that *Hook Module* intercepts, which parameters of these APIs that *Hook Module* parses, and which audit types the parsed parameters are leveraged to perform. The details of hooking are described in Section 4.

TABLE 2
APIs and Parameters Related to the Sensing Analysis
in *senDroid*

Sensors Type	Flow Audit	Frequency Audit	Duration Audit	Invoker Audit
Graphic Sensors	Preview	ioctl:DQBUF	ioctl:STREAMON/OFF	ioctl: BINDER_WRITE_READ: _Service name and sender_euid
	Video		ioctl: BINDER_WRITE_READ: ICamera: START/STOP_RECORDING /IMediaRecorder: START/STOP	
	Photo	-	-	
Audio Sensors		ioctl: BINDER_WRITE_READ: ICamera:TAKE_PICTURE	ioctl: BINDER_WRITE_READ: IAudioRecord/ IMediaRecorder: START/STOP	
Location Sensors		ioctl:READI_FRAMES	ioctl: BINDER_WRITE_READ: ILocationManager: REQUEST/REMOVE_UPDATES	
Standard Sensors		msg_q_rcv:REPORT_POSITION	ioctl: BINDER_WRITE_READ: SensorEventConnection: SET_EVENT_RATE	
			ioctl: BINDER_WRITE_READ: SensorEventConnection: ENABLE/DISABLE	

Parameter Parser. The submodule of *Hook Module* is a set of functions that we implement to substitute the system calls that are related to sensor accessing. These functions intercept all the communication requests sent from an application to sensor devices or Binder. When a system call is intercepted, *Parameter Parser* parses the parameters carried by the system call. According to the parsing result, the system calls are delivered to *Usage Generator* for further analysis.

Usage Generator. Leverages a set of audit policies to generate the sensing usage reports. Each entry to report sensing usage is defined as a vector, which includes the accessing time, data size and sensor accessor. The information delivered from *Parameter Parser* is sporadic. So we need to gather the information to create sensing usage reports, which are stored in a local database. These reports are available to be accessed by *Sensing Monitor*.

3.4 Sensing Monitor

Sensing Monitor is an application that allows a user to analyze the sensing usages generated by *Hook Module*. *Sensing Monitor* can visualize the reports of the sensing usages by applications or by sensors in a comprehensive manner. Fig. 3 shows one user interface of *Sensing Monitor*. A graph of location sensors usage is presented to the user, where the red curve denotes the overall usage over the time. The usage of each application is represented by curves in different colors.

4 SENDROID: IMPLEMENTATION

We leverage the graphic sensors as an example to illustrate the implementation of *senDroid*:

Fig. 4 shows the interposition of *senDroid* in the data flow of `startPreview`. The application process does not actually communicate with the camera device. Instead, the interaction with the camera device is implemented in the *mediaserver* process. The application process informs the *mediaserver* process of its request through Binder. When receiving the request sent by the application process, the camera service and camera driver pass the request to the camera device. The camera service also returns the execution results via Binder.

The interaction between the *mediaserver* process and Binder indicates that the request is sent by which application and is delivered to which service. The interaction

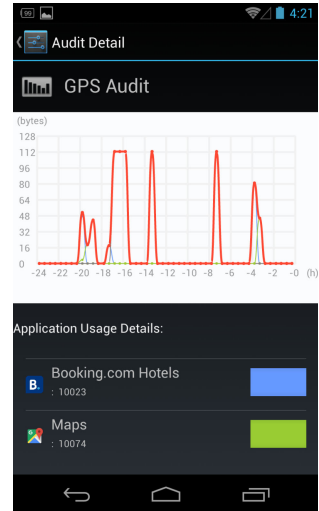


Fig. 3. Visual demonstration of sensing usages. The report includes the total usages for all applications, and the individual usage of each application.

between the *mediaserver* process and the camera device indicates what and how many data the application obtains. These two interactions are the choke points of the data flow and are where *senDroid* interposes.

4.1 Interception Between Service and Binder

We leverage *senDroid* to intercept interactions between the service and Binder. Concretely, the *Hook Module* hooks the `ioctl` called by Binder. Then, the *Parameter Parser* parses the `read_buffer` in the structure `binder_write_read`, and finds out all the `BR_TRANSACTION` commands (illustrated in Section 2.2.1). Furthermore, we extract the receiver and the requester from the first entry of buffer in the structure `binder_transaction_data` and the field `sender_euid`, respectively. Finally, we identify the operation according to the field code.

4.2 Interception Between Service and Devices

4.2.1 Graphic Sensors

senDroid intercepts the `ioctl` calls with the commands `VIDIOC_STREAMON` and `VIDIOC_STREAMOFF` to learn when an application opens and closes cameras. When starting preview, the camera driver calls `ioctl` with the command `VIDIOC_DQBUF`, and the corresponding data buffer

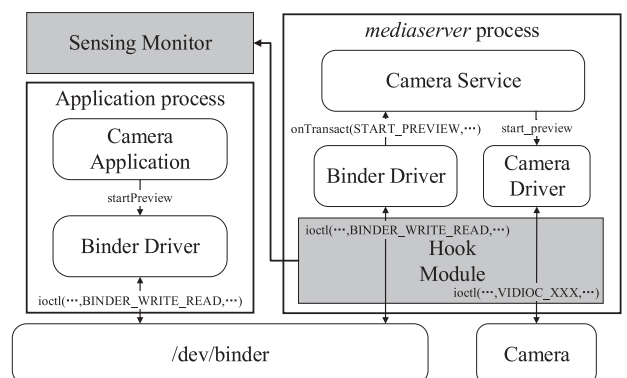


Fig. 4. Interposition of *senDroid* in the data flow of `startPreview`. The shadowed blocks are designed in *senDroid*.

is a structure named `v4l2_buffer`. This structure contains variables, from which we can infer the size of a graphic frame. At the driver layer, the preview, the photo, and the video are all graphic frames so that we cannot pick out a photo or a piece of video from the stream of graphic frames. Fortunately, we can demarcate the stream according to the Binder requests sent by the applications. The `ioctl` calls called by Binder sending request to `ICamera` with the code `TAKE_PICTURE` indicates that an application is taking photos. Similarly, Binder request sent to `ICamera` or `IMediaRecorder` with the code `START/STOP_RECORDING` extracts a piece of video from the stream of frames. It is noteworthy that an application can take pictures or record videos using `PreviewCallback` instead of `takePicture` or `MediaRecorder`. `senDroid` monitors Binder request sent to `ICamera` with the code `SET_PREVIEW_CALLBACK_FLAG` to learn whether a `PreviewCallback` is registered before the preview starts or during previewing.

4.2.2 Audio Sensors

`senDroid` intercepts the `ioctl` calls invoked by the audio driver with the command `SNDRV_PCM_IOCTL_READI_FRAMES`. We calculate the size of audio record from the corresponding data buffer named `snd_xferi`. ALSA only has the command `SNDRV_PCM_IOCTL_START`, with no command `SNDRV_PCM_IOCTL_STOP`. So we obtain the start and end time of the audio record by the approach described in Section 4.1. Concretely, we capture the `ioctl` calls called by Binder where the service name is `android.media.IAudioRecord` or `android.media.IMediaRecorder` with the code either `START` or `STOP`. We use the time of these calls as the start and end time of the audio record.

Considering that the `MediaRecorder` can be utilized to record audio as well as video, and before using `MediaRecorder` to record audio or video, applications are required to invoke `setAudioSource` and `setVideoSource` respectively, we distinguish these two usages by monitoring the invocation of `setAudioSource` and `setVideoSource`. The corresponding codes in the `ioctl` called by Binder are `SET_AUDIO_SOURCE` and `SET_VIDEO_SOURCE` respectively.

4.2.3 Location Sensors

When the location information is sent via the message queue mechanism, which is illustrated in Section 2.2.2, `senDroid` hooks the message receiving calls called by the GPS driver and parses the message received from the GPS device. We determine the operation specified by the message according to the message id. `senDroid` intercepts the messages with the message id `REPORT_POSITION`, which indicates that the received data are geographic positions reported to a location requester. We intercept the `ioctl` calls where the service name is `android.location.ILocationManager` to determine the start and end time of the tracking of GPS according to the code `REQUEST_LOCATION_UPDATES` and `REMOVE_UPDATES`.

4.2.4 Standard Sensors

As mentioned in Section 2.2.2, standard sensors have no open source driver on Nexus 4. So we cannot learn how

standard sensors driver communicates with corresponding device. However, `senDroid` can intercept the `ioctl` calls called by Binder that sending request to `SensorEventConnection` with the code `ENABLE_DISABLE`. This Binder request carries two parameters: a standard sensor handle and a Boolean (`TRUE` presents enabling and `FALSE` presents disabling). So we can know which standard sensor is activated and when it is activated or deactivated. Moreover, Binder request with code `SET_EVENT_RATE` contains a parameter that indicates the access frequency of standard sensors application. Because the data size of one standard sensor record is negligible. So although we cannot obtain the exact data of standard sensors, we argue that activated time interval and access frequency of standard sensors are crucial and sufficient to audit the sensing based on the standard sensors.

4.3 Prototype Setup

We use the Nexus 4 with Android 4.2.2 as our experiment platform. When we set up our prototype, we need to root the device first, and then leverage the existing DLL injection approach² to inject into the target processes (`mediaserver`, `system_server` and application processes that access sensors³), load the function substitution codes into the target processes and execute the function substitution. Then, we leverage the implementation in [27] to accomplish the function substitution. The function substitution code goes through the memory map of the target processes and load each ELF file to substitute the functions we concerned, e.g., `ioctl` for our own functions. After deploying `senDroid`, all four categories of sensors, including camera, microphone, GPS, and standard sensors, will be audited. The sensing usage is logged in XML format for subsequent analysis in `Sensing Monitor`.

5 SENDROID: EVALUATION

In order to evaluate the accuracy and overhead of `senDroid`, we set up the application dataset which consists of 540 applications downloaded from the top free chart of `Google Play Store` in April, 2016. We only keep the applications which require access to sensors monitored by `senDroid` while remove the others. Of the 540 applications, 429 applications require access to one or more of the four categories of sensors. Further, We ignore applications which have internal failure or whose sensor related functionalities cannot be reached due to geological or system version restriction. For camera, microphone and GPS, we determine whether an application requires access to these three categories of sensor information based on the permissions it requests. For standard sensors, we determine that an application requires access to standard sensor data if it declares `<uses-feature>` tag for `android.hardware.sensor.*`.

We randomly select three ones for each type of sensors under the auditing of `senDroid`. All evaluation experiments

2. <https://github.com/shutup/libinject2>

3. The camera service and audio service are launched in `mediaserver`, the location service and sensors service are launched in `system_server`. We cannot guarantee whether an application directly accesses sensors via JNI, so we inject into every application process.

TABLE 3
Comparison Between the Time of Accessing Sensors
Observed Manually and the Time of Accessing
Sensors Recorded by *senDroid*

Action	App	Mean start time $ \Delta $ (s)	Mean stop time $ \Delta $ (s)
Taking pictures	PicsArt	0.00	-
	Poshmark	2.00	-
	Perfect365	0.66	-
Taking video	Zoosk	1.00	0.50
	Tango	0.50	0.50
	ooVoo	1.00	0.50
Recording audio	Tom Loves Angela	0.00	0.00
	Tango	0.00	0.00
	Talking Tom	0.00	0.00
Requesting location	CM Security	0.50	0.00
	Expedia	0.00	0.00
	360 Security	0.00	0.00
Reading standard sensors	Temple Run 2	4.00	2.50
	Bowling Kings	2.00	2.50
	Traffic Racer	3.00	3.00

are run on a Nexus 4 with 2 GB RAM running Android 4.2.2.⁴

5.1 Deviation of Data Reported by *senDroid*

5.1.1 Experiment Results

Table 3 shows the experiment results for applications which use camera to take pictures or take videos, use microphone to record audio, use GPS or network to request locations, and access standard sensors. We record the start time and end time of each *Action* on each *App*. We repeat the above operation three times. Then, in Table 3, *Mean start time* $|\Delta|$ refers to the average value of start time difference between the value recorded by a tester and by *senDroid*. And *Mean stop time* $|\Delta|$ refers to the average value of end time difference between the value recorded by a tester and by *senDroid*.

Applications listed in Table 3 respectively provide users with the functions of taking pictures, taking videos, recording audios, requesting locations, and reading standard sensors. We record the time when we trigger the actions and the time when *senDroid* detects that the application executes the actions. Considering the time from UI operations to the time the application actually executes the actions, the deviation is acceptable. As is shown Table 3, *senDroid*'s deviation for detecting access to standard sensor data is larger than that of other sensors. Because games will usually play animations before and after the game starts or ends, so it is more difficult to infer the time when applications start or stop requesting standard sensor data.

4. Although the new version of Android has many new features, the technical details of sensor access are almost same. We, thus, use this version as our experiment platform. Furthermore, these popular applications may run this version of Android, and be empirical studied according to sensor usages in the next section.

TABLE 4
Comparison Between the Actual Length of Videos and Audios
and the Length Which *senDroid* Records

		Lengths(s)			
		Actual	Average recorded by <i>senDroid</i> (Differences)		
			Device 1	Device 2	Device 3
video	10.00	11.30(1.30)	11.31(1.31)	11.34(1.34)	
	30.00	31.32(1.32)	31.28(1.28)	31.31(1.31)	
	60.00	61.31(1.31)	61.38(1.38)	61.33(1.33)	
	120.00	121.24(1.24)	121.31(1.31)	121.31(1.31)	
audio	10.00	10.21(0.21)	10.21(0.21)	10.20(0.20)	
	30.00	30.19(0.19)	30.19(0.19)	30.21(0.21)	
	60.00	60.21(0.21)	60.25(0.25)	60.23(0.23)	
	120.00	120.23(0.23)	120.24(0.24)	120.23(0.23)	

5.1.2 Accuracy of Recorded Video & Audio Length

We developed an evaluation application to measure the length of videos and audios which *senDroid* records, and compared them with the actual lengths. We leveraged the `setMaxDuration` API of the class `android.media.MediaRecorder` to set the length of the videos and audios. After the recording reaches the length we set, `MediaRecorder` will stop the recording. Here, we deploy *senDroid* on three different Nexus 4 in order to explore whether the deviation differs on different devices. For each value of length, we ran the test for five times and calculated the average length recorded by *senDroid*.

As is shown in Table 4, the lengths which *senDroid* record are about 1.3 seconds and 0.2 second more than the actual lengths of the video and audio, respectively, and the deviation is almost the same on different devices. Moreover, as the lengths of the video and audio increase, the deviation almost remains unchanged. According to our investigation, the deviation may be caused by the overhead of inter-process communication or the preprocessing and postprocessing of the record. Since the deviation is consistent for the same type of sensor across different devices and different lengths of recordings, we ignore the deviation in the length of recordings reported by *senDroid*.

5.2 Audit of Taking Pictures from Preview Frames

As we described in Section 4.2.1, an application can process the provided preview data in the `onPreviewFrame` callback. This means that applications can take pictures or record videos without calling the `takePicture` or `MediaRecorder` API by starting camera preview and taking screenshot programmatically.

To better evaluate the effectiveness of *senDroid* with this case, we evaluated *senDroid* with an application called *Spy Camera HD*.⁵ According to *Spy Camera HD*'s description on *Google Play Store*, it allows users to secretly take photos without any shutter sound and camera preview on the phone screen. Users can instruct *Spy Camera HD* to take pictures by shaking the phone, whistling or setting a timer. After reverse-engineering *Spy Camera HD*, we confirm that *Spy Camera HD* actually takes pictures in the

5. <https://play.google.com/store/apps/details?id=com.fullapps.spycamera>

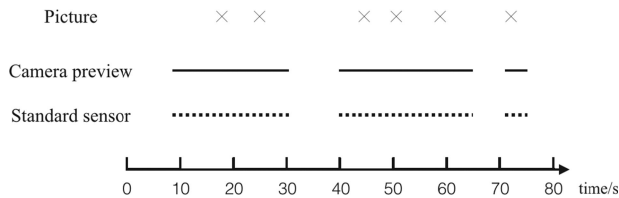


Fig. 5. The timestamps of pictures taken by Spy Camera HD and the time durations of Spy Camera HD's usage of sensors detected by *senDroid*.

onPreviewFrame callback it registers by processing the content data of the preview frame which is provided as an argument when the onPreviewFrame callback is called. Even on devices which shutter sound is enabled forcibly, capturing the provided preview data in onPreviewFrame instead of calling takePicture directly will not trigger any shutter sound. In this way, Spy Camera HD can take pictures without people's awareness.

We used Spy Camera HD to take several pictures by shaking the phone and matched the time when the pictures were taken based on the pictures' timestamps with the duration of camera preview and the duration of application requesting standard sensor data which *senDroid* records. The result is shown in Fig. 5. The first row shows the time which Spy Camera HD took a picture based on the timestamp of the saved picture file. For example, the first picture was taken at the 18th second. The second and third rows show the time duration when Spy Camera HD started the camera preview and requested standard sensor data respectively. For example, the first duration of camera preview started at the 9th second and ended at the 30th second, which also matches the first duration of Spy Camera HD's requesting for standard sensor data. During the test, we took five pictures, and there were three durations when Spy Camera HD started camera preview and requested standard sensor data at the same time. We can conclude from Fig. 5 that all pictures taken with Spy Camera HD fall in the camera preview duration and standard sensor data request duration which *senDroid* records.

Since *senDroid* will not only record applications' calling the takePicture API, it will also detect applications which override the onPreviewFrame callback and record the duration which applications start the camera preview, these two tricky methods of taking pictures can be detected by *senDroid*. With *senDroid*, users can have a complete overview of how applications access the camera and be aware of potential suspicious usage of camera and other sensors.

5.3 Audit of Dynamically Loaded APK

In February 2016, *Alipay*, backed by Chinese e-commerce giant Alibaba, was accused of taking pictures and recording audios secretly with its Android client [2]. As discovered by Twitter user typcn [28], the code logic of *Alipay* for Android's suspicious behavior consists of a file with the extension *.so* downloaded from a remote server via the Internet. The *.so* file is actually an executable APK file, which will be run as a plugin inside *Alipay* for Android. Since the *.so* file will be checked for updates periodically, the suspicious behavior was quickly removed quietly after the incident became popular on the Internet without users' updating the host *Alipay* for Android application itself.

TABLE 5
Macrobenchmark Results

Action	w/o <i>senDroid</i> (s)	w/ <i>senDroid</i> (s)	Overhead
Picture	2.38(0.09)	2.58(0.14)	8.05%
Video	115.80(0.46)	115.84(0.14)	0.04%
Audio	103.32(0.07)	103.37(0.08)	0.04%

We built an application which emulates the dynamic APK loading feature of *Alipay* for Android based on the open source project ACDD [29]. This application can download APK files as plugins from a remote server, extract DEX files from the APK files and finally run the code logic in the APK files using *DexClassLoader*, which is similar to the claimed mechanism leveraged by *Alipay* for Android. Plugins are compiled with a patched *aapt* tool, so the resource IDs of plugins will not conflict with those of the host application, and the host application can distinguish resources from different plugins by reading the first byte of the resource ID. The *mInstrumentation* variable of the *android.app.ActivityThread* class is also hooked, so that when components such as activities and services in the plugins are launched, corresponding resources are loaded. The host application which we built is a dummy application which only loads and runs APK files from a remote server, while a dynamically loaded APK file contains the logic of taking pictures and recording audios. As we tested, *senDroid* successfully detected the usage of camera and microphone in the dynamically loaded APK.

5.4 Performance Evaluation

5.4.1 Performance of Sensor Operations

To measure the performance overhead of calling the sensor APIs incurred by *senDroid*, we built a test application to take pictures, record videos, and record audios successively, and compared the time which it takes to perform the same operations with *senDroid* installed or not. For the operation of taking pictures, we takes 10 pictures successively. The measurement includes the time from executing the first taking picture action by calling the takePicture API to the calling of the onPictureTaken callback for the last picture. Note that, to better reflect the actual overhead of calling the camera APIs incurred by *senDroid*, all pictures are dropped without saving to the disk, so the time of disk I/O is eliminated. For the operation of recording video and audio, the customized application records 10 pieces of video/audio with a length of 10 seconds. This measurement includes the time from preparing recording the first video/audio to the calling of the onInfo callback with a "what" code of MEDIA_RECORDER_INFO_MAX_DURATION_REACHED for the last video/audio. Due to restriction of the *MediaRecorder* API, video and audio cannot be recorded by *MediaRecorder* without saving, so the results include the time of disk I/O. Since both the takePicture API and the *MediaRecorder* API are IPC-based APIs which requires *senDroid* to extract information from the *ioctl* system call, the evaluation results can truly reflect *senDroid*'s performance overhead in the worst cases.

All tests were performed on a same newly flashed Nexus 4 with 2 GB RAM running Android 4.2.2 after a reboot in the same environment. Table 5 shows the results. The numbers

TABLE 6
AnTuTu Benchmark Results (Larger Numbers in Cells Imply More Efficient Performance for Test Cases)

	Without <i>senDroid</i>	With <i>senDroid</i>	Overhead
RAM	5058.4(218.9)	5040.6(167.7)	0.35%
CPU mathematics	3600.4(58.8)	3558(25.4)	1.18%
CPU common use	3692.8(305.0)	3442.6(116.3)	6.78%
CPU multi-core performance	2511.8(67.7)	2526(30.5)	-0.57%
UX data security	1270.4(72.5)	1291.4(8.3)	-1.65%
UX data processing	549.6(12.2)	541.4(7.9)	1.49%
UX strategy games	754.8(20.1)	750.4(11.5)	0.58%
UX image process	259.4(5.7)	260.2(3.9)	-0.31%
UX I/O performance	1487(19.5)	1527.6(33.6)	-2.73%
Overall	19184.6(212.985)	18938.2(200.3)	1.28%

in parentheses indicate the expected range of values with a confidence interval of 95 percent. *senDroid* adds approximately 8.05, 0.04 and 0.04 percent overhead to taking pictures, recording videos and recording audios, respectively. The additional overhead can be attributed to storing the sensor usage to database. Thus, we can conclude that the performance overhead imposed by *senDroid* is negligible.

5.4.2 AnTuTu Benchmark

Further, we want to know whether *senDroid* will also introduce negligible overhead to the overall system or not. We use a popular Android benchmarking tool: AnTuTu [30] to compare the system performance with and without *senDroid*. The average benchmark results of five tests for a Nexus 4 running Android 4.2.2 with and without *senDroid* respectively are shown in Table 6. Similarly, the number in parentheses indicating the expected range of values with a confidence interval of 95 percent. We can conclude from Table 6 that *senDroid* also imposes negligible overhead with only 1.28 percent on the overall system performance.

6 SENDROID: EMPIRICAL STUDY

6.1 Experiment Setup

In this section, we conducted an extensive, empirical study on the sensor usage in real applications from popular markets. Before the study, we collected applications from *Wandoujia*, *360* and *Google Play Store* in September and October

TABLE 7
Application Category Composition of *Wandoujia* Dataset

Category	Number	Category	Number
COMMUNICATION	30	UTILITY	29
EDUCATION	30	GAMES	28
BEAUTY&BABY	30	TRANSPORTATION	28
MUSIC	30	TOOLS	28
LIFESTYLE	30	SHOPPING	27
PHOTOGRAPHY	30	PRODUCTIVITY	25
NEWS&MAGAZINES	30	VIDEO	23
PERSONALIZATION	29	SOCIAL	15
FINANCE	29	HEALTH&FITNESS	3
TRAVEL	29		

There are 503 Applications from 19 Categories.

TABLE 8
Application Category Composition of *360* Dataset

Category	Number	Category	Number
FINANCE	30	COMMUNICATION & SOCIAL	28
PHOTOGRAPHY	30	WALLPAPER	27
LIFESTYLE	30	BUSSINESS	27
GAMES	30	EDUCATION	22
NEWS&MAGAZINES	29	SYSTEM SECURITY	21
SHOPPING	29	HEALTH&MEDICAL	15
MAPS&TRAVEL	29	MUSIC&VIDEO	14

There are 361 Applications from 14 Categories.

2016, among which *Wandoujia* and *360* are predominant application markets in China. We mainly picked top free applications of each category, and the category distributions of applications in each application market are shown in Tables 7, 8 and Table 9 respectively. Note that, there exists an applications' overlap between these three markets, which means that a same application can be tested twice or three times. Especially, there are 119 applications being tested both in *Wandoujia* and *360*. Basically, we focus our study on the accesses to sensors of the applications in the following two phases:

- *Launch Phase*: Once an application is launched, it may detect the availability of the concerned sensors or gather sensors data for initialization and cause a large amount of accesses to sensors. We install and start all applications in dataset on a smartphone one by one. Here, we define a *Launch Phase* interval for 2 minutes just after each application started, and *senDroid* will audit the accesses to sensors during this interval and write the report in a log file.
- *Silent Phase*: We also evaluate the accesses to sensors when an application is running in the background. The study for *Silent Phase* can help to reveal the

TABLE 9
Application Category Composition of *Google Play Store* Dataset

Category	Number	Category	Number
TOOLS	35	NEWS_AND_MAGAZINES	25
COMICS	34	PRODUCTIVITY	23
PERSONALIZATION	34	HEALTH_AND_FITNESS	22
BOOKS_AND_REFERENCE	32	MEDICAL	22
WEATHER	31	GAMES	20
BUSINESS	29	LIFESTYLE	20
SOCIAL	29	TRAVEL_AND_LOCAL	20
COMMUNICATION	27	ENTERTAINMENT	19
EDUCATION	27	LIBRARIES_AND_DEMO	19
PHOTOGRAPHY	27	MEDIA_AND_VIDEO	19
MUSIC_AND_AUDIO	26	SPORTS	17
SHOPPING	26	FINANCE	16
TRANSPORTATION	26		

There are 625 Applications from 25 Categories.

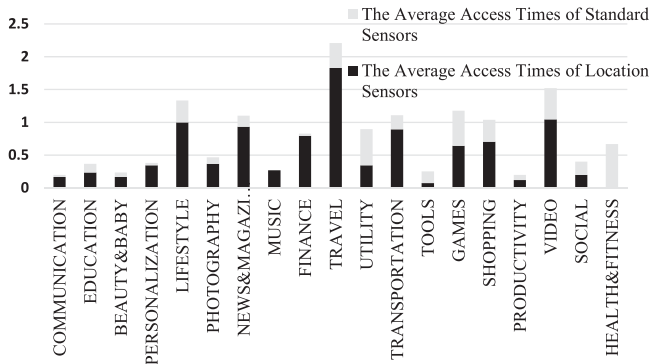


Fig. 6. The average access times during *Launch Phase* to each type of sensors in each category of *Wandoujia* market.

applications' malicious use or abuse of sensors in current prevailing application markets. In this phase, any sensor usage recorded by *senDroid* will be regarded as suspicious usage. Due to the memory limitation of the tested Android device, we run 16 applications in the background at the same time for about 24 hours, thus to make sure that we would get accurate and enough information about sensor usages.

Note that, although it should be more efficient to run evaluation tests on emulators, audio sensors and standard sensors are unfortunately disabled and cannot be emulated on emulators [31], [32]. Thus we must conduct the experiments manually on real devices.

6.2 Experiment Results of Empirical Study

We studied applications downloaded from *Wandoujia*, *360* and *Google Play Store* respectively from October, 2016 to January, 2017. Because the numbers of applications in different markets and categories are not uniform, to eliminate the interference, we calculate the percentage of applications in the results.

6.2.1 Launch Phase

For *Wandoujia*, we got 2,209 records of sensor accesses, among which 396 records were generated during *Launch Phase*. The accesses mainly concentrate on GPS and standard sensors: There are 117 (23.3 percent) applications under 18 different categories out of 503 valid samples access location sensors in *Launch Phase*. The accesses to location sensors are very widespread. As is shown in Fig. 6, the

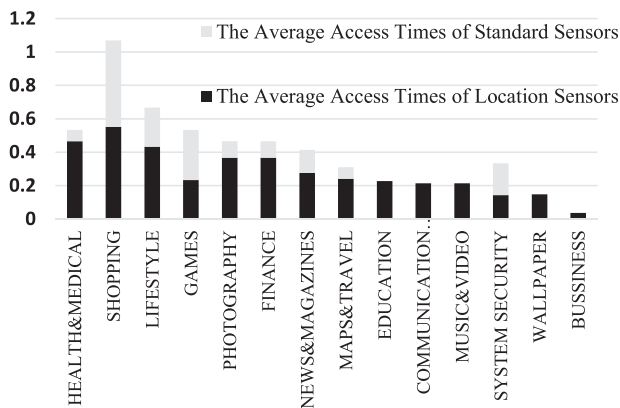


Fig. 7. The average access times during *Launch Phase* to each type of sensors in each category of *360* market.

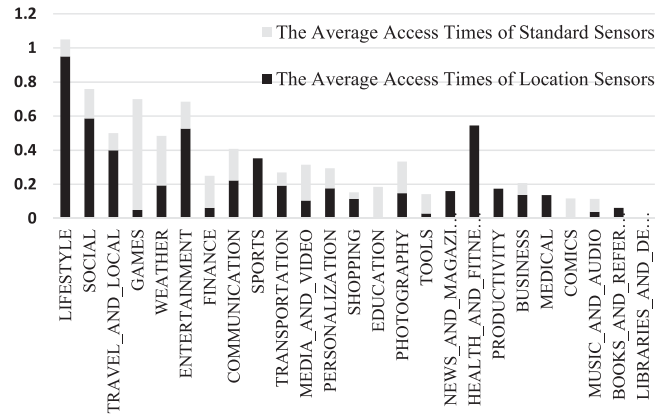


Fig. 8. The average access times during *Launch Phase* to each type of sensors in each category of *Google Play Store* market.

applications under the categories of *Lifestyle*, *Travel* and *Shopping* are most active to access location sensors in this phase. Besides, about 12.3 percent of applications try to access various kinds of standard sensors during the *Launch Phase*. Differing from location sensors, standard sensors are typically accessed by applications of *Health&Sports*, *Games* and *Videos*. We also have an inspection into the accessing frequency of each kind of standard sensors during this phase. Among all the six types of standard sensors, accelerometer is the most popular one and the number of its related records is far more than the records of other types of standard sensors. Note that, few application try to fetch the graphic sensors data or audio sensors data in this phase, and the only four applications are basically scan tools or shooting tools, which depends their main functions on the use of camera preview.

For *360*, there are 103 and 42 applications accessed location sensors and standard sensors respectively, while only one accessed graphic sensors and one accessed audio sensors. Comparing with *Wandoujia*, the ratio of location sensors usages in *360* is a little bit higher while the ratio of standard sensor usage is lower. We calculate the average access times to each type of sensors in each category respectively as shown in Fig. 7. Similarly, the applications in *Shopping* and *Lifestyle* categories still behave actively not only on accessing location sensors but also on accessing standard sensors. Over 55 percent of the applications in *Shopping* category triggered location sensors access, and 51.7 percent of *Shopping* applications accessed standard sensors.

At last, for the applications in *Google Play Store*, the overall accessing during *Launch Phase* is relatively less than applications from the above two markets. During the *Launch Phase*, only 79 out of 625 applications accessed location sensors, and 56 applications were recorded with standard sensors' accessing history. Similarly, the accesses to accelerometer are the most frequently among all types of standard sensors, while there are no access record for light and proximity sensor data. An overall distribution of categories that accessed sensors during *Launch Phase* is shown in Fig. 8.

6.2.2 Silent Phase

We carefully reviewed all the records and filtered out the applications which accessed sensors during the *Silent Phase* for each markets respectively, since any usage in this phase

TABLE 10
Percentages of Sensor Usages in Three Different Markets

	Overall	Launch Phase	Silent Phase
<i>Wandoujia</i>	33.40%	28.82%	13.92%
<i>360</i>	37.67%	34.90%	16.62%
<i>Google Play Store</i>	22.40%	19.36%	5.44%
Total	29.82%	26.32%	11.01%

can be regarded as suspicious usages. Still, no matter in which one of the markets, accesses to location sensors prevail and the next is standard sensors. There are 13.9 percent applications in *Wandoujia* accessing sensors during *Silent Phase* while for *360* the ratio is 16.6 percent. However, for *Google Play Store*, the ratio is far more lower. Only 5.4 percent of applications in it tried to access sensors. Interestingly, we find that quite a portion of applications access the sensors with a certain time interval and a fixed accessing period. This interval can be one hour, or two hours and the access may last for 2 seconds or even longer. We further study the reason why it happens and we would give a deeper discussion about suspicious usages in *Silent Phase* in Section 6.3.

6.2.3 Comparison and Summary

After the data collection and statistics, we analyze the results of both *Launch Phase* and *Silent Phase*, and make a comparison of the overall sensor usages in these three markets in Table 10. We can read from the table that sensors are used widely in *Wandoujia* and *360*, even in *Silent Phase*. And the usages of location sensors and standard sensors in different stages are also given in Table 11. Besides, both the usage of graphic sensors and the usage of audio sensors are less frequent, and the percentages are less than 1.0 percent, no matter in *Launch Phase* or *Silent Phase*. For *Google Play Store*, there is an observable gap between its usage ratio and the ratio of the other two markets. Considering that the applications we collected are applications on the top free charts, and the dominant user group of these three markets are different, we can reasonably conclude that the users of *Google Play Store* may be more likely to download the sensor-friendly applications.

Sensors are widely used in various categories of applications. In *Wandoujia*, over 60 percent of the applications in *Travel* and *Lifestyle* ever accessed sensors during our experiments, no matter in *Launch Phase* or *Silent Phase*. While nearly half of the applications under *Shopping*, *Video* and *Games* used sensors more or less. In *360*, *Health&Medical*, *Shopping*, *Lifestyle* and *Games* are in the front rank. At last, in *Google Play Store*, applications in *Lifestyle*, *Social* and *Travel&Local* behaved actively. Although the category lists of each application market are not totally the same, we can still find that certain categories of applications behave more actively than the others. For example, *Lifestyle* and *Games* are two representative categories in which sensors are extensively accessed.

6.3 Analysis and Case Study

As we mentioned above, any sensor access in *Silent Phase* is regarded as suspicious usage. In order to find out the

TABLE 11
Percentage of Location Sensors and Standard Sensors in Different Markets in Different Stages

	GPS	Standard Sensors
<i>Wandoujia</i>	23.26% / 12.33%	12.13% / 1.99%
<i>360</i>	28.25% / 14.40%	11.63% / 3.60%
<i>Google Play Store</i>	12.64% / 4.64%	7.68% / 2.66%
Chinese App Markets	25.35% / 13.19%	11.92% / 2.66%
Overall	20.01% / 9.60%	10.14% / 2.01%

The data is presented in the format of usage percentage in *Launch Phase* / usage percentage in *Silent Phase*.

applications which generated a huge amount of sensor data in our experiment, especially during *Silent Phase*, and how the sensor data would be used, we drill down into every application that accesses sensors in *Silent Phase*. Specifically, we reverse-engineer the APK files and explore the source codes to figure out what the accessed sensor data are used for. Then we compare the usages with the description and privacy policy of applications.

6.3.1 Sensor Access Patterns and Third-Party Libraries

After reverse-engineering the APK files, we find that most of them access location sensors by third-party ad libraries or analytic libraries. In order to trace the data flow, we first scan the source codes for the sensor-related API and record the third-party package names if any. Next, combining with the access records and the applications' basic information we collected beforehand, we judge if there is any abuse or misuse of sensor data.

Among all applications in *Wandoujia*, there are 70 applications which accessed sensors during *Silent Phase*, and 53 of them accessed in a relatively high frequency. Because part of the applications applied anti-reverse engineering technologies or the tools we used to reverse-engineer exist deficiencies, we can only get the source codes of 42 applications. Most of the applications accessed sensors according to an obvious pattern. For example, there are 16 applications accessed location service every 4,850 seconds. This is because all of them contain a third-party library named *cn.jp.push.android*, which includes the codes that will call the location sensors related services. In other cases, there are 6 applications contain the location sensor calling methods both in themselves and the third-party libraries they employ, and only one application's sensor related codes are not contained in third-party libraries. Therefore, the overall percentage of sensor accessing in third-party libraries is 83.33 percent.

For *360* market, there are 48 applications accessed sensors frequently during *Silent Phase*. After successfully reverse-engineering 28 of them, we find that, similarly to the case in *Wandoujia*, 6 applications present an accessing pattern with a fixed accessing interval of 4,850 seconds, which is caused by *cn.jp.push.android* as well. 75 percent of the applications are found containing the sensor accessing codes in their third-party libraries. This phenomenon further verifies the fact that third-party libraries are blamed for the frequent suspicious accessing of sensors during *Silent Phase*.

While in *Google Play Store*, we do not find a uniform pattern since the accesses during *Silent Phase* are fewer than

TABLE 12
Top Third-Party Libraries Which Appears the Most Frequently in Our Experiments

Market	Package Name of Third-Party Libraries	Times
Wandoujia	<i>cn.jpush.android</i>	22
	<i>com.tencent.map</i>	14
	<i>com.baidu.location</i>	9
	<i>com.qq.e.comm.managers.status</i>	8
	<i>com.loc</i>	7
	<i>com.amap.api.location</i>	7
	<i>com.aps</i>	7
	<i>com.alipay.mobilesecuritysdk.model</i>	6
	<i>com.tencent.mm.sdk.platformtools</i>	6
	360	<i>cn.jpush.android</i>
<i>com.baidu.location</i>		12
<i>com.tencent.map</i>		7
Google Play Store	<i>com.flurry.sdk</i>	9
	<i>gms</i>	9

that of the above two markets. There are 34 applications accessed sensors during *Silent Phase* and only 18 of them accessed in a relatively high frequency. However, we still find that 12 applications accessed sensors because of the related codes in third-party libraries' packages, which account for 66.67 percent of the total. And the other 6 applications contain the sensor accessing codes both in themselves and their containing third-party libraries.

From all the studies above, we find that 77.27 percent of the accessing records are caused by the third-party libraries. Table 12 shows the top third-party libraries which are used in our experiments. It reasonably explains the reason why the applications belongs to *Finance* and other categories, which seems to have no need to access sensors frequently, would generate so many accessing records in our experiment: although the applications themselves may have no intention to fetch any sensor data while running in the background, the third-party libraries call the sensor related service, which disobey the original motivation of the applications' developers. This possibility of sensor usages in third-party libraries can hardly be noticed by the developers and thus they would not mentioned in applications' descriptions, which would mislead the users and bring about the risk of privacy leakage. As a result, we argue that a responsible application should not only give a detailed description of the application's functionalities, but also list the sensor permissions required by both the application and its contained third-party libraries if any. Moreover, the description should clarify that which kind of sensor will be used under what circumstances, thus to eliminate the worries about privacy information leakage from users.

6.3.2 Suspicious Case Study

We further traced the data flow and summarized the concrete usage scenarios of the sensor data. Basically, we focus on the applications themselves how to use the data rather than the third-party libraries. After the analysis, we summarize the usage scenarios of location sensors' data as following:

- 1) Sending the collected sensor data to a specific server, but the further use can not be traced.

TABLE 13
The Application Packages Which Access the Location Sensor Data During *Silent Phase* and Their Corresponding Usage Scenario

package name	scenario	package name	scenario
<i>com.letv.android.client</i>	4	<i>com.nd.android.pandahome2</i>	5
<i>cn.ledongli.lidl</i>	3	<i>viva.reader</i>	1
<i>com.jsmmc</i>	2	<i>com.cmcm.whatscall</i>	1
<i>com.vlocker.locker</i>	2	<i>com.handmark.expressweather</i>	3
<i>com.cleanmaster.mguard_cn</i>	3	<i>com.pingenie.screenlocker</i>	2

- 2) Using the collected data as a keyword for other information, e.g., pulling down the local weather information after accessing the location sensor and getting the location information.
- 3) Storing the collected data locally on device for the possible future use.
- 4) Exporting the sensor data to the system logs.
- 5) Doing nothing at all.

Based on the different usage scenarios, we categorize the application packages according to their usage in Table 13. Besides, we also have an insight into the usages of standard sensor data. Basically, we can divide the ways they are used into as following: (1) Testing the device's rotation angle; (2) Testing the device's shaking; (3) Testing the brightness surrounding. And a detailed result is shown in Table 14.

At last, we investigated all the descriptions displayed on the application markets of applications mentioned above, and found that none of them notices the possible sensor usage. Thus we want to emphasize the importance of application's description, and insist that the developers should be responsible for clarifying the third-party libraries used in the application and which of them would possibly cause the use of sensors.

7 DISCUSSION

7.1 Coverage

senDroid can implement sensing audit by (1) intercepting the IPC between applications and services or (2) intercepting the communication with sensor drivers. Considered in terms of generality, all four categories sensors can be audited by the first approach because the IPC mechanisms are uniform in different version of Android. The first approach can be detoured by attacks that directly access sensor drivers without communicating with the sensor service. So with respect to robustness, *senDroid* can intercept the communication with sensor drivers to defense the attacks or detect suspicious accesses to sensors.

TABLE 14
The Application Packages Which Access the Standard Sensor Data During *Silent Phase* and Their Corresponding Usage Scenario

package name	usage scenario
<i>com.cleanmaster.mguard_cn</i>	3
<i>cld.navi.mainframe</i>	1
<i>com.lashou.groupurchasing</i>	2
<i>com.taobao.ju.android</i>	1
<i>com.ubercab</i>	1

The coverage of the second approach depends on the source code of sensor drivers we can analyze. For graphic sensors and audio sensors, Linux kernel provides standard device drivers. Thus, by making the interception according to the standard invocation mode, *senDroid* is capable of auditing all accesses to graphic and audio sensors. For location sensors, with no device driver provided by Linux kernel, *senDroid* only supports sensing audit on devices that use Qualcomm GPS driver. Because the driver of standard sensors on Nexus 4 is close-source, *senDroid* only implements the sensing audit by intercepting the IPC between applications and services. We argue that, with the support of manufacturers of sensors, the location sensors and standard sensors can be audited by intercepting the communication with sensor drivers.

7.2 Suspicious Usage Patterns

senDroid provides users with the detailed and visual sensing usage reports. Users can infer from the usage reports which stealthy application is accessing sensors at an unexpected time or with an abnormal data size. For professional users, the reports of *senDroid* may offer more technical details, and help the professional users find more suspicious accesses to sensors. However, *senDroid* does not support the suspicious usage pattern recognition now. This absence could bring up a burden on common users, especially, when the sensing relevant applications are widely used.

A possible solution is to collect plenty of sensor data usage patterns of malwares. Then we can mine classification rules by applying machine learning on the usage patterns. Based on the suspicious usage patterns, we can then improve *Sensing Monitor* to be an application which can automatically identify the suspicious usage of running applications, and alert users if the alert rules are applied.

7.3 Bypassing *senDroid*

senDroid implements the sensing audit by intercepting the data flow and the interception relies on substituting function pointers in the ELF file of a target process. A malicious application can apply the similar interception to bypass *senDroid* either by substituting the sensor-related system calls for its own implementation, or by breaking down *senDroid* completely.

However, it is challenging for the attacker because the malicious application must call `ptrace` to attach to the target process before function substitution. So if *senDroid* is attached to the target process before the attacker, according to the documentation of `ptrace` [33], the later attaching will cause error and thus fail. *senDroid* is attached to target processes as soon as the system is launched. Even when attacker makes the attachment before *senDroid*, we can be informed of the abnormal behavior and give the user a warning.

8 RELATED WORK

Hooking is a technique for inserting codes into a system call for alteration, and it can be used to intercept the applications' requests, thus to realize the sensor-related behavior check. Xu et al. developed *Aurasium* in [27]. *Aurasium* can keep on monitoring any security or privacy violations in Android OS. Basically, *Aurasium* realizes the enforcement of its security policies by hooking into

applications processes. *FireDroid* [34] is another work that relies on hooks. It intercepts the system calls to identify if an application is executing dangerous actions at runtime. Similarly, *FireDroid* performs security checks on applications and enforces security policies. *DeepDroid* [35], a dynamic enterprise security policy enforcement scheme on Android devices, also dynamically hooks system processes in order to find details of applications' requests for a fine-grained access control. *Boxify* [36] presents a concept for full-fledged app sandboxing on stock Android and it also aims at enforcing established security policies. Although the core technique used in these works are similar to ours, however, no matter in *Aurasium*, *FireDroid*, *DeepDroid* or *Boxify*, they aim at realizing a policy-based security, which means they design the system with hooks in order to apply some specific security policies to prevent users from attacks, while our work is different from others by focusing on auditing sensor access in Android system-wide.

Besides hooking, another technique is to modify the existing Android sensor framework to intercept the sensor data flow. Xu et al. [37] proposed a sensor management framework, called *SemaDroid*, based on the *SemaHooks*, which are not real hooks but are codes embedded within the existing components in the Android framework. *SemaDroid* provides the users with capacity of monitoring the sensor usage of installed applications and also provides a fine-grained and context-aware access control. Basically, *SemaDroid* focuses on supporting of context-aware and quality-of-sensing based access control policies though it offers the possibility of auditing the sensor usage by giving a sensor usage report as an individual application as well.

ipShield [38] is a framework that monitors sensor accessed by an application and assesses the privacy risk of the sensor access. Besides, it also gives recommendations of sensor configurations to users and supports sensor related access control actions. *Scippa* [39], an extension to the Android IPC mechanism, provides provenance information required to effectively prevent recent attacks such as confused deputy attacks. Heuser et al. [40] proposed the Android Security Modules (ASM) framework, which provides a programmable interface for defining new reference monitors for Android. Particular reference monitor can be developed to monitor sensor access. These designs can be bypassed in the access mode of JNI, where the accesses to sensors do not pass the Android framework layer. Different from these works, *senDroid* implements the interception at the device driver layer without any modification to the Android framework. So *senDroid* can be widely deployed and is capable of detecting potential attacks that bypass the Android framework. *senDroid* can not only detect which application is accessing which sensor but also quantify how many data the application accessed. Furthermore, the hook-based method can audit applications even when they bypass the Android framework via JNI invocations. The implementation and evaluation show that *senDroid* is effective and efficient to audit sensing in the Android platform.

Enck et al. [5] proposed a dynamic taint tracking and analysis system, named *TaintDroid*, that is capable of simultaneously tracking multiple sources of sensitive data. *DroidTrack* is a method proposed in [41] for tracking and visualizing the transmission of privacy information and

preventing its leakage. *AppIntent* [42], a framework analyzes data diffusion to help an analyst to determine whether the data diffusion is user intended or not. These mechanisms can track the target data, including personal privacy. The tracking reports of them are very limited while *senDroid* reveals more details of the sensing operations.

Wijesekera et al. [43] did a field study to find how often applications access protected resources when users are not expecting it on Android platform and they hooked the permission-checking APIs in data collection step. However, in their study, they focus on the permissions or resources of connectivity, location, view, and so on. In *senDroid*, we implement a meticulous study focusing on sensors on the Android platform.

9 CONCLUSION AND FUTURE WORK

This paper proposes *senDroid*, which may be used to monitor and analyze the sensing operations in the Android platform. To the best of our knowledge, it is the first work to design a tool to audit the sensing in the Android platform without the changes of the source codes of the Android framework. *senDroid* leverages a hook-based method to implement the interception of the sensor related API calls. According to the results of our conducted experiments, *senDroid* can efficiently gather the data of all four categories of sensors in the Android platform, i.e., graphic sensors, audio sensors, location sensors, and standard sensors, with high accuracy. In addition, *senDroid* can work even when suspicious or malicious codes are dynamically loaded from server sides or bypass the middleware of Android via JNI calls. Next, *senDroid* can monitor the suspicious behaviors where an application extracts the graphic data from the preview frames of a camera to silently take photos. This behavior can bypass the alert of the shutter voice which is mandatorily open in some countries, such as China. The performance report shows that the [0.04-8.05] percent overheads for different operations are promising. Finally, our empirical study on applications from real markets shows that it is very high-frequent for popular applications in *Wandoujia*, *360* and *Google Play Store* to access sensors. We also find that many applications access location sensors and standard sensors when applications access location sensors and standard sensors when applications are running in the background and third-party libraries are blamed for the continuous accesses, but the developers do not declare the usage in the description or privacy policy of the application. To the best of our knowledge, it is also the first empirical study on the dynamic usages of sensors of Android applications.

In our future work, we plan to conduct more experiments for a large scale applications to discover more cases of suspicious and malicious usages of sensors. In addition, we will improve our analysis tool to automatically report the malicious usages based on our gathered data and other relevant data of applications, such as application descriptions. Last but not least, we will support the further sensing audit under the supports of device manufactures.

ACKNOWLEDGMENTS

This paper is supported by NSFC (Grant No. 61572136, 61472358, 61370080), the National Program on Key Basic Research (2015CB358800), and the Shanghai Innovation Action Project (Grant No. 16DZ1100200, 18511103600).

We thank all anonymous reviewers for their insightful comments. We are now sharing all source codes of *senDroid* on GitHub (<https://github.com/letitb/senDroid>).

REFERENCES

- [1] L. Atzoria, A. Ierab, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, pp. 2787–2805, 2010.
- [2] R. Liu, "Alipay dismisses accusation it violated user-privacy by snapping photos." [Online]. Available: <http://www.allchinatex.com/alipaydismisses-accusation-it-violated-user-privacy-by-snappingphotos/>, Accessed on: 2016.
- [3] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion," in *Proc. 6th USENIX Workshop Hot Topics Security*, 2011, vol. 11, pp. 9–9.
- [4] Y. Michalevsky, D. Boneh, and G. Nakibly, "GyropPhone: Recognizing speech from gyroscope signals," in *Proc. 23rd USENIX Security Symp.*, 2014, pp. 1053–1067.
- [5] W. Enck, et al., "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Trans. Comput. Syst.*, vol. 32, no. 2, 2014, Art. no. 5.
- [6] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *Proc. 4th Int. Conf. Commun. Syst. Netw.*, 2012, pp. 1–9.
- [7] S.-W. Lee and K. Mase, "Activity and location recognition using wearable sensors," *IEEE Pervasive Comput.*, vol. 1, no. 3, pp. 24–32, 2002.
- [8] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, 2008, pp. 323–336.
- [9] T. Watanabe, M. Akiyama, and T. Mori, "Routedetector: Sensor-based positioning system that exploits spatio-temporal regularity of human mobility," in *Proc. Usenix Conf. Offensive Technol.*, 2015, p. 6.
- [10] D. Currie, "Shedding some light on voice authentication," 2009.
- [11] A. Al-Haiqi, M. Ismail, and R. Nordin, "On the best sensor for keystrokes inference attack on android," *Procedia Technol.*, vol. 11, pp. 989–995, 2013.
- [12] R. Spreitzer, "Pin skimming: Exploiting the ambient-light sensor in mobile devices," in *Proc. 4th ACM Workshop Security Privacy Smartphones Mobile Devices*, 2014, pp. 51–62.
- [13] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile device identification via sensor fingerprinting," *CoRR*, vol. abs/1408.1416, 2014, [Online]. Available: <http://arxiv.org/abs/1408.1416>
- [14] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "AccelPrint: Imperfections of accelerometers make smartphones trackable," in *Network and Distributed System Security Symposium*. New York, NY, USA: Citeseer, 2014.
- [15] H. Wang, D. Lymberopoulos, and J. Liu, "Sensor-based user authentication," in *Wireless Sensor Networks*. Berlin, Germany: Springer, 2015, pp. 168–185.
- [16] T. Schreiber, "Android binder," *A Shorter, More General Work, but Good for an Overview Binder*. 2011. [Online]. Available: <http://www.nds.rub.de/media/attachments/files/2012/03/binder.pdf>
- [17] Android, "Android interface and architecture." [Online]. Available: <https://source.android.com/devices/>, Accessed on: 2015
- [18] B. Nguyen, "Linux dictionary," 2003.
- [19] eLinux.org, "Executable and linkable format (elf)." [Online]. Available: [http://elinux.org/Executable_and_Linkable_Format_\(ELF\)](http://elinux.org/Executable_and_Linkable_Format_(ELF)), Accessed on: 2015
- [20] E. Bendersky, "Position independent code (PIC) in shared libraries." [Online]. Available: <http://eli.thegreenplace.net/2011/11/03/positionindependent-code-pic-in-shared-libraries/>, Accessed on: 2015
- [21] J. Shewmaker, "Analyzing DLL injection," *GSM Presentation*, 2006.
- [22] P. Padala, "Playing with ptrace, part II," *Linux J*, vol. 104, 2002, Art. no. 4.
- [23] Z. Xu, K. Bai, and S. Zhu, "Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proc. 5th ACM Conf. Security Privacy Wireless Mobile Netw.*, 2012, pp. 113–124.
- [24] R. Schlegel, K. Zhang, X.-Y. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundcomber: A stealthy and context-aware sound trojan for smartphones," in *Proc. 2nd Netw. Distrib. Syst. Security Symp.*, 2011, vol. 11, pp. 17–33.

- [25] Z. Zhang, P. Liu, J. Xiang, J. Jing, and L. Lei, "How your phone camera can be used to stealthily spy on you: Transplantation attacks against android camera service," in *Proc. 5th ACM Conf. Data Appl. Security Privacy*, 2015, pp. 99–110.
- [26] W. Enck, D. Ocateau, P. McDaniel, and S. Chaudhuri, "A study of android application security," in *Proc. USENIX Security Symp.*, 2011, vol. 2, Art. no. 2.
- [27] R. Xu, H. Saïdi, and R. Anderson, "Aurasium: Practical policy enforcement for android applications," in *Proc. Presented Part 21st USENIX Security Symp.*, 2012, pp. 539–552.
- [28] typcn, "typcn on twitter." [Online]. Available: https://twitter.com/typcn_com/status/701706390218231808, Accessed On: 2016–05–10.
- [29] bunnyblue, "Acdd." [Online]. Available: <https://github.com/bunnyblue/ACDD>, accessed: 10-May-2016
- [30] AnTuTu, "Antutu benchmark 3d- android apps on google play." [Online]. Available: <https://play.google.com/store/apps/details?id=com.antutu.benchmark.full>, Accessed on: 10-May-2016
- [31] Android, "Audio capture." [Online]. Available: <https://developer.android.com/guide/topics/media/audio-capture.html>, Accessed on: 2016
- [32] Android, "Sensors overview." [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-practices, Accessed on: 2016
- [33] M. Kerrisk, "ptrace(2)." [Online]. Available: <http://man7.org/linux/man-pages/man2/ptrace.2.html>, Accessed on: 2016
- [34] G. Russello, A. B. Jimenez, H. Naderi, and V. D. M. Wannes, "Firedroid: Hardening security in almost-stock android," in *Proc. Comput. Security Appl. Conf.*, 2013, pp. 319–328.
- [35] X. Wang, K. Sun, Y. Wang, and J. Jing, "Deepdroid: Dynamically enforcing enterprise policy on android devices," in *22nd Ann. Netw. Distrib. Syst. Secur. Symp.*, NDSS 2015, San Diego, California, USA, Feb. 8–11, 2015.
- [36] M. Backes, S. Bugiel, C. Hammer, O. Schranz, and P. Von Styp-Rekowsky, "Boxify: Full-fledged app sandboxing for stock android," in *Proc. Usenix Security Symp.*, 2015, pp. 691–706.
- [37] Z. Xu and S. Zhu, "SemaDroid: A privacy-aware sensor management framework for smartphones," in *Proc. 5th ACM Conf. Data Appl. Security Privacy*, 2015, pp. 61–72.
- [38] S. Chakraborty, C. Shen, K. R. Raghavan, Y. Shoukry, M. Millar, and M. Srivastava, "ipShield: A framework for enforcing context-aware privacy," in *Proc. 12th USENIX Conf. Netw. Syst. Des. Implementation*, 2014, pp. 143–156.
- [39] M. Backes, S. Bugiel, and S. Gerling, "Scippa: System-centric IPC provenance on android," in *Proc. Comput. Security Appl. Conf.*, 2014, pp. 36–45.
- [40] S. Heuser, A. Nadkarni, W. Enck, and A. R. Sadeghi, "ASM: A programmable interface for extending android security," in *Proc. 23rd USENIX Conf. Security Symp.*, 2014, pp. 1005–1019.
- [41] S. Sakamoto, K. Okuda, R. Nakatsuka, and T. Yamauchi, "DroidTrack: Tracking and visualizing information diffusion for preventing information leakage on android," *J. Internet Serv. Inf. Security*, vol. 4, no. 2, pp. 55–69, 2014.
- [42] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, and X. S. Wang, "Appintent: Analyzing sensitive data transmission in android for privacy leakage detection," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 1043–1054.
- [43] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov, "Android permissions remystified: A field study on contextual integrity," in *Proc. Usenix Conf. Security Symp.*, 2015, pp. 499–514.

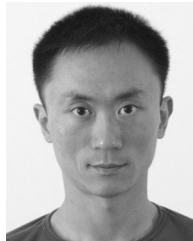


Weili Han (M'08) is a full professor in the Software School at Fudan University. His research interests are mainly in the fields of data systems security, access control, digital identity management. He is now a member of the ACM, SIGSAC, IEEE, and CCF. He received his Ph.D. degree at Zhejiang University in 2003. Then, he joined the faculty of the Software School at Fudan University. From 2008 to 2009, he visited Purdue University as a visiting professor funded by China Scholarship Council and Purdue University.

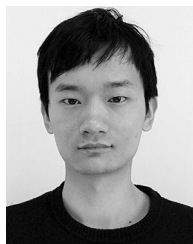
He serves in several leading conferences and journals as PC members, reviewers, and an associate editor. He is a member of the IEEE.



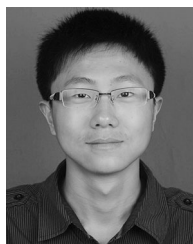
Chang Cao is a graduate student majored in Computer Software and Theory at Fudan University. She received her B.S. degree from Fudan University in 2016. She is currently a member of the Laboratory for Data Analytics and Security. Her research interest mainly includes security-related access control on mobile devices.



Hao Chen is a full professor in the Department of Computer Science at the University of California, Davis. He received his Ph.D. degree at the Computer Science Division at the University of California, Berkeley, and his BS and MS from Southeast University. His research interests are computer security, machine learning, and mobile computing.



Dong Li is currently a software engineer in Works Applications Co., Ltd. He received his B.S. degree in Computer Science from Beijing Institute of Technology in 2014 and the M.S. degree in Computer Software and Theory from Fudan University in 2017. His research interests are mainly in the fields of access control on Android.



Zheran Fang is currently a software engineer at Microsoft. He received his M. S. degree in Computer Science from Fudan University. He was a member of the Laboratory of Cryptography and Information Security, Software School, Fudan University. His research interests mainly included information security and policy based management.



Wenyan Xu (M'07) is a full professor in the College of Electrical Engineering at Zhejiang University. She received her B.S. degree in Electrical Engineering from Zhejiang University in 1998, her M.S. degree in Computer Science and Engineering from Zhejiang University in 2001, and the Ph.D. degree in Electrical and Computer Engineering from Rutgers University in 2007. Her research interests include wireless networking, smart systems security, and IoT security. Dr. Xu received the NSF Career Award in 2009 and was

selected as a young professional of the thousand talents plan in China in 2012. She was granted tenure (an associate professor) in the Department of Computer Science and Engineering at the University of South Carolina in the U.S. She has served on the technical program committees for several IEEE/ACM conferences on wireless networking and security. She has published over 60 papers and her papers have been cited over 3000 times (Google Scholar). She is a member of the IEEE.



X. Sean Wang is a full professor in the School of Compute Science at Fudan University, Shanghai, China. He received his Ph.D. degree in Computer Science from the University of Southern California in 1992. Before joining Fudan University in 2011, he was the Dorothean Chair Professor in Computer Science at the University of Vermont. His research interests include data systems and data security. He is a member of ACM and senior member IEEE.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.

The Overhead from Combating Side-Channels in Cloud Systems Using VM-Scheduling

Nahid Juma¹, Jonathan Shahan¹, Khalid Bijon¹, and Mahesh Tripunitara¹

Abstract—Recent work suggests that scheduling, with security as a consideration, can be effective in minimizing information leakage, via side-channels, that can exist when virtual machines (VMs) co-reside in clouds. We analyze the overhead that is incurred by such an approach. We first pose and answer a fundamental question: is the problem tractable? We show that the seemingly simpler sub-cases of initial placement and migration across only two equal-capacity servers are both intractable (NP-hard). However, a decision version of the general problem to which the optimization version is related polynomially is in NP. With these results as the basis, we make several other contributions. We revisit recent work that proposes a greedy algorithm for this problem, called Nomad. We establish that if $P \neq NP$, then there exist infinitely many classes of input, each with an infinite number of inputs, for which a decrease in information leakage is possible, but Nomad provides none, let alone minimize it. We establish also that a mapping to Integer Linear Programming (ILP) in prior work is deficient in that the mapping can be inefficient (exponential-time), and therefore does not accurately convey the overhead of such an approach that, unlike Nomad, actually decreases information leakage. We present our efficient reductions to ILP and boolean satisfiability in conjunctive normal form (CNF-SAT). We have implemented these approaches and conducted an empirical assessment using the same ILP solver as prior work, and a SAT solver. Our analytical and empirical results more accurately convey the overhead that is incurred by an approach that actually provides security (decrease in information leakage).

Index Terms—Cloud computing, information security, cross-VM side-channels, computational complexity

1 INTRODUCTION

CLOUD computing has become an important paradigm by offering flexibility and cost-effectiveness to both providers of infrastructure and services, and their clients. An aspect of cloud computing is virtualization, which enables providers to host virtual machines (VMs) of multiple clients on the same physical infrastructure. Indeed, multiple VMs may be co-resident, that is, reside on the same server. While co-residency has benefits, such as economies of scale, it gives rise to a serious security threat. Information may leak from a victim client's VM to a malicious co-resident VM via a side-channel attack. In such an attack, the attacker runs a VM on the same server as that of the victim's VM and takes advantage of a shared physical component in order to extract information about the victim. For example, the attacker may retrieve the victim's cryptographic key by observing the activity of the processor cache. Prior work exposes several such co-residency side-channels in shared cloud environments (see, for example, [1], [2], [3], [4], [5], [6], [7], [8], [9]).

A straightforward solution to this security issue is hard-isolation: preclude co-residency and give each client dedicated hardware. This is unfavourable because it reduces efficient use of resources. Several other defences that do not

involve the complete elimination of co-residency have been suggested by prior work (see, for example, [10], [11], [12]). Many of these defences suffer from at least one of the following drawbacks. First, they entail changes to existing deployments and applications (see Section 7 and [13] for a discussion). Second, they are catered specifically to known cross-VM side-channel attacks. Therefore, as new attacker capabilities are unveiled, more changes may have to be made. It is desirable to have defences that are general across a broad spectrum of side-channel attacks and immediately deployable with no or only few modifications to existing cloud hardware and software.

With this objective in mind, scheduler-based defences have been proposed in recent work [13]. These can be thought of as a realization of the moving target defence philosophy to mitigate side-channels [14]. The mindset is that security is a factor which should be considered when deciding how VMs are placed on servers, rather than as an after-thought.

A scheduler has several mechanisms it can employ to maximize security. Two that have been explored in recent prior work by Moon et al. are the following [13]. A scheduler can be careful with where it places a newly arrived VM thereby limiting information leakage from or to it. Migration of existing VMs is yet another mechanism, with the intent that a victim VM does not leak too much information to a co-resident malicious VM. In Fig. 1, we illustrate a scheduler-based defence. In the figure, a scheduler places and migrates client VMs in a manner that minimizes information leakage between them.

Scheduler-based defences appear to have promise as they offer several advantages. First, they focus on the root cause

- N. Juma, J. Shahan, and M. Tripunitara are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. E-mail: {ndawood, jmshahan, tripunit}@uwaterloo.ca.
- K. Bijon is with Composure.AI (formerly called MosaixSoft Inc.), Los Altos, CA 94022. E-mail: khalid@composure.ai.

Manuscript received 13 July 2017; revised 26 Sept. 2017; accepted 15 Dec. 2017. Date of publication 8 Jan. 2018; date of current version 18 Mar. 2020.
(Corresponding author: Nahid Juma.)

Digital Object Identifier no. 10.1109/TDSC.2018.2790932

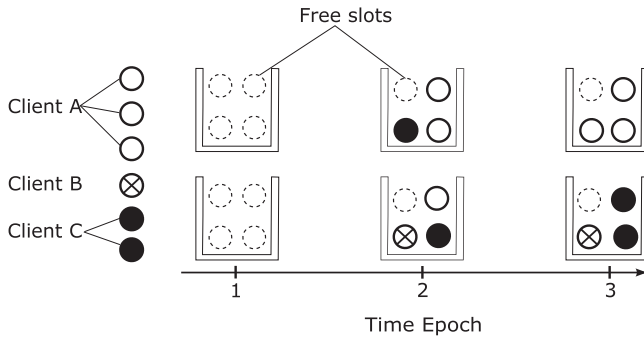


Fig. 1. Scheduling actions that can mitigate information leakage. Time is discretized into periods called *epochs*. A scheduler places VMs, such as the ones that arrive in Epoch 1 and are placed in Epoch 2. It also migrates VMs, which results in a new placement for Epoch 3. We use this example in several sections of the paper. In particular, the placement in Epoch 2 may or may not be considered to suffer from more information leakage than the one in Epoch 3, depending on the model for leakage that we adopt.

of side channels, i.e., co-residency, and are agnostic to the specific side-channel attack used. This makes them robust against unforeseen side-channels that meet certain conditions [13]. Second, they require no changes to the cloud provider’s hardware, client applications, and hypervisors and can be deployed “out of the box” as they require only changing the VM placement and/or scheduling algorithm deployed by the cloud provider. While we acknowledge that there may be other defences, e.g., shielded execution [15], that also offer these advantages to some extent, our focus in this work is on scheduler-based defences.

An important contribution of Moon et al. [13] is four models that quantify information leakage in co-resident settings. These allow for the security-sensitive scheduling problem to be posed precisely as an optimization problem—given a migration budget, we seek a placement of VMs on servers that minimizes information leakage. That work explores three approaches to solve the problem, namely a mapping to ILP, an initial greedy algorithm called Baseline Greedy and a final greedy algorithm whose associated software is called Nomad.

Our primary intent is to analyze the overhead of introducing such security-sensitive scheduling to cloud systems. We make three sets of contributions.

Contribution Set 1: Tractability. We first pose and answer a fundamental question: is the underlying problem tractable?¹ A reason that this question is interesting is that we know that the cloud scheduling problem for other objectives, such as utilization, is intractable, i.e., **NP**-hard [16]. Is it the same for the objective of decreasing information leakage?

We first establish an upper-bound for the hardness of the problem. We show that a decision version of the problem, to which the optimization version reduces polynomially, is in **NP** (Section 3.1). We show also that the problem is indeed **NP**-hard and is therefore **NP**-complete (Section 3.2). We do this by showing that even for the simpler sub-case in which

1. Some prior work uses the terms “tractable” and “intractable” to refer to an *approach* rather than the underlying *problem* for which the approach is proposed as a solution [13]. For clarity and in consistency with customary usage, we use the terms “tractable” and “intractable” for a problem, and “efficient” and “inefficient” for an approach.

there are only two servers of equal capacity, deciding whether there exists a placement that meets a certain threshold of information leakage is **NP**-hard. A lower-bound hardness for that problem is a lower-bound for our more general problem for so-called open systems—settings in which VMs may come and go over time. We then establish that even the special case of the problem for closed systems, under the $\langle R, C \rangle$ model for information-leakage, is **NP**-hard (Section 3.3). In closed systems, there already exists a placement of VMs, and there are no arrivals or departures. This special case is of interest because it is the only case supported by the implementation of Nomad that has been made available to us (see Section 6).

Given the above results, and the customary assumption, that $\mathbf{P} \neq \mathbf{NP}$, no efficient (polynomial-time) algorithm exists for minimizing information leakage. We explore ways to mitigate this intractability (Section 3.4). In particular, we discuss our investigations into tractable sub-cases of the problem, and whether the problem may demonstrate certain kinds of optimal substructure.

Contribution Set 2: Prior Approaches. Our second set of contributions is an analysis of the three approaches taken by Moon et al. to address security-sensitive scheduling [13].

That work asserts that mapping the problem to ILP is not promising in practice—even for modestly sized problem instances, the approach takes longer than a day for a commercial ILP solver on a reasonable computer. We analyze their mapping and observe that while it does appear to be a reduction, it can be inefficient² (Section 4.1)—there exists an encoding of placement for which the output of the mapping from that work is exponential in the size of its input. The fact that an underlying decision problem is in **NP** means that an efficient (polynomial-time computable) reduction to ILP exists. Because that work does not make use of an efficient reduction, it does not meaningfully characterize the overhead incurred with an ILP approach.

We then observe that their starting point in the design of a heuristic algorithm, Baseline Greedy, is, in the worst-case, more inefficient than it needs to be (Section 4.2). Given that a decision version of the underlying problem is in **NP**, we know that there exists an algorithm that can find the optimal solution given time $\Theta(2^n)$, where n is the size of the input.³ We observe that baseline greedy runs in time $\Theta(n!)$ in the worst-case. And $2^n = o(n!)$, that is, the function $n!$ is an upper-bound for 2^n that is not tight.

Their final design, Nomad, runs in time polynomial (quadratic) in the size of the input. Then, given that the underlying problem is **NP**-hard and the customary premise $\mathbf{P} \neq \mathbf{NP}$, we ask a natural question: what does Nomad give up for efficiency?

We show that Nomad provides no security for a large number of input instances (Section 4.3). Under the customary premise, $\mathbf{P} \neq \mathbf{NP}$, there are an infinite number of input

2. We distinguish correctness from efficiency in this context. A reduction is a mapping that satisfies a particular “if and only if” property. It may not be computable efficiently.

3. We use $\Theta(\cdot)$ and $o(\cdot)$ in a manner that is customary in computing [17]. The set of functions $\Theta(f(n))$ comprises those that are bounded asymptotically from above and below by $f(n)$. The set $o(f(n))$ comprises those functions that are bounded asymptotically from above by $f(n)$, but not tightly.

instances for which Nomad provides no reduction in information leakage whatsoever, let alone minimize it. Indeed, in the worst case, Nomad suffers the maximum information leakage possible, even when it is unconstrained by a migration budget, and even though there exists a placement that results in zero information leakage.

A natural consequent question one may ask is, can Nomad, or a similarly efficient algorithm, be modified to identify such instances yet retain its efficiency? The answer is ‘no,’ unless $\mathbf{P} = \mathbf{NP}$ (Section 4.3).

Contribution Set 3: True Overhead. In our third set of contributions, we address the question as to what the overhead really is, for an approach that indeed minimizes information leakage. Our approach to answering the question is to first recognize that, as the problem is \mathbf{NP} -hard, we cannot hope to have an algorithm which is efficient for all inputs. However, a related decision-version is in \mathbf{NP} . Consequently, we have designed two approaches to the problem.

In one of our approaches, we reduce the problem to boolean satisfiability in conjunctive normal form (CNF-SAT) with the intent of adopting a SAT solver as an oracle. And in the other, we reduce the problem to ILP with the intent of adopting an ILP solver as an oracle. Both CNF-SAT and ILP are known to be \mathbf{NP} -complete [18]. Our reduction to ILP differs from that of prior work in that ours is computable in polynomial-time, and, as a consequence, the size of its output is polynomial in the size of the input.

We have implemented both of our approaches. Also, we have conducted an empirical assessment of our approaches, and Nomad. We have validated some of our analytical observations on Nomad against its implementation. The reason we implemented both reductions is that we know from practice that solvers for the two problems may not demonstrate the same performance on all input instances. Thus, even though both CNF-SAT and ILP are \mathbf{NP} -complete, it is interesting to investigate whether one of those approaches outperforms the other for different classes of input instances.

Our implementations are discussed in Section 5, and we make empirical observations in Section 6. Our empirical observations are consistent with the main themes of this work. That is, we present empirical results on (a) the overhead incurred, in practice, by approaches that actually minimize information-leakage, and, (b) validation of our analytical insights on the shortcomings of Nomad that we discuss in Section 4. On (a), for example, we find that while approaches based on reduction to ILP and CNF-SAT do not scale as well as Nomad, their scalability appears to be much better than previously reported [13]—for example, for a cluster of 40 servers, an approach based on reduction to ILP takes 4 minutes, and not longer than a day as previously reported (see Section 6).

Layout. We describe the problem we address in Section 2. This is the same problem that is addressed by Moon et al. [13]. Our results on the computational complexity of the problem are in Section 3. In Section 4, we describe and analyze the three approaches (i.e., a mapping to ILP, Baseline Greedy and Nomad) taken by Moon et al. [13]. We discuss our approaches that are based on reductions to CNF-SAT and ILP in Section 5. We present empirical results for our approaches and Nomad in Section 6. We address related work in Section 7 and conclude with Section 8. The appendix

TABLE 1
The Symbols Used for Calculating Total Information Leakage

Symbol	Meaning
$v_{c,i}$	The i^{th} VM of client c .
$Y_{c,i,c',i'}(e')$	An indicator variable, i.e., it takes the value 0 or 1 only. It takes the value of 1 if $v_{c,i}$ and $v_{c',i'}$ are co-resident on a server during an epoch e' . Otherwise it takes the value 0.
$I_{c,i,c',i'}(e, \Delta)$	The VM-to-VM information leakage from $v_{c,i}$ to $v_{c',i'}$ over the sliding window of epochs $[e - \Delta, e]$. This quantity is calculated as
	$\sum_{e' \in [e - \Delta, e]} Y_{c,i,c',i'}(e')$
$I_{c,c'}(e, \Delta)$	The client-to-client information leakage from c to c' over the sliding window of epochs $[e - \Delta, e]$. This quantity is dependent on the presence (R, C) or absence (NR, NC) of replication and collaboration.
$I_{total}(e, \Delta)$	The total information leakage over the sliding window of epochs $[e - \Delta, e]$, under one of the four leakage models, $\{R, NR\} \times \{C, NC\}$. This quantity is calculated by aggregating the client-to-client leakage across all pairs of clients.

contains proofs and discussions that we were unable to include in the main body on account of page-limits. The appendix is available online [50].

2 MINIMIZING INFORMATION LEAKAGE

In this section, we describe the problem posed by Moon et al. [13]. The problem is: given (i) a set of clients, each having a number of VMs, (ii) a set of servers, each having a capacity expressed as a number of VMs, and (iii) a migration budget, that limits the number of VM migrations that can occur, what is a placement of VMs on servers that minimizes information leakage? Our exposition is similar to that of Moon et al. [13], with changes for clarity only.

Setup. Time is discretized into periods called *epochs*. A VM is a virtual machine—a unit of execution, similar to a process in a conventional operating system. A VM, when run, may last several epochs. A *client* is associated with a set of VMs; every VM belongs to a client. The set of VMs that belong to a client can change over time. A *server* is a machine on which a VM runs; the VM is said to be *hosted* by that server. A server is associated with a capacity—it is able to host a certain number of VMs only, at any given time. VMs may be *migrated*—hosted by a different server in a later epoch. A potential victim client has some secret information in her VMs. A malicious client attempts to steal the secret of a victim client via information-leakage across VMs that can occur when the VMs are *co-resident*, i.e., hosted by the same server. We do not know beforehand which VMs are potential victims, and which are malicious. In the leakage models we consider, clients do not share secrets, and a client’s secret may be present in more than one of its VMs. Fig. 1 shows three servers and clients, and six VMs across those three clients, and hosting over three epochs. These notions are further explained by Table 1, which presents the symbols we use in our calculations of information-leakage.

We quantify information as a certain number of bits. We premise that merely owing to co-residency of a victim and

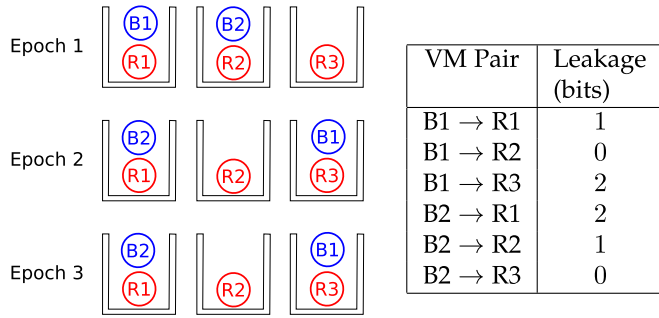


Fig. 2. Example used in the prose to show how client-to-client leakage is calculated under all four models. The leakage from B1 to R1 is 1 bit over the three epochs because B1 is co-resident with R1 for one out of the three epochs.

a malicious VM, information leaks from the former to the latter. Information that leaks from a VM to another aggregates over the time that the two VMs are co-resident. In order to aggregate information leakage as a function of co-residency over time, we consider a *sliding window* of the most recent Δ epochs. This can model secrets, such as cryptographic keys, that are refreshed periodically; information the adversary gathered in previous sliding windows is no longer useful to her. We assume that during an epoch, a VM leaks 1 bit to another that is co-resident. This can be generalized to the case that the amount of leakage is some constant, k bits. The value k can then be used to model the time-length of an epoch. The amount of information leaked from a victim to an adversary is proportional to the time their VMs are co-resident with one another. For example, if a VM of Client A is co-resident with a VM of Client B for 2 epochs, then the total information leaked from Client A's VM to Client B's VM over 2 epochs is 2 bits.

Replication and Collusion. Replication means that information is duplicated across a victim client's VMs. This is potentially advantageous for an attacker because when an attacker VM is co-resident with multiple victim VMs, under the assumption that the attacker VM extracts different bits of information from each victim VM, the rate of leakage to the attacker increases. For example, in Fig. 1, for the placement in Epoch 2, with replication, Client C (black), on server 1, gains twice as many bits from Client A (white) than it would without replication.

Collusion means that a malicious client's VMs can collaborate with one another. This is potentially advantageous for the attacker because when multiple attacker VMs are co-resident with a victim VM, under the assumption that each attacker VM extracts different bits from the victim VM, the rate of leakage to the attacker, as a whole, increases. In Fig. 1, for the placement in Epoch 2, with collusion, Client A (white), on server 1, gains twice as much information from Client C (black) than it would without. As in prior work, we denote the presence of replication and collusion by R and C respectively, and the absence by NR and NC respectively. This leads to four possible leakage models: $\langle NR, NC \rangle$, $\langle R, NC \rangle$, $\langle NR, C \rangle$, $\langle R, C \rangle$.

Information Leakage Models. We now describe how client-to-client leakage is calculated under the four models, using the example in Fig. 2.

For the $\langle NR, NC \rangle$ case, the information leakage from client c to client c' is the maximum per-VM-pair information leakage across all pairs of VMs. This is the information which leaks to the adversary VM that is co-resident with the same victim VM for the largest number of epochs

$$\langle NR, NC \rangle : I_{c,c'}(e, \Delta) = \max_i \max_{i'} I_{c,i,c',i'}(e, \Delta).$$

In Fig. 2, the maximum VM-to-VM leakage from the blue client (B) to the red client (R) is 2 bits, so the leakage from B to R under $\langle NR, NC \rangle$ is 2 bits over the three epochs.

For the $\langle R, NC \rangle$ case, there is a cumulative effect across victim VMs. The information leakage from client c to client c' is determined by the adversary VM that has extracted the most information

$$\langle R, NC \rangle : I_{c,c'}(e, \Delta) = \max_{i'} \left[\sum_i I_{c,i,c',i'}(e, \Delta) \right].$$

In Fig. 2, R1 obtains the most information, a total of $1+2=3$ bits, so the leakage from B to R under $\langle R, NC \rangle$ is 3 bits.

For the $\langle NR, C \rangle$ case, there is a cumulative effect across adversary VMs. The information leakage from client c to client c' is determined by the victim VM that has leaked the most information

$$\langle NR, C \rangle : I_{c,c'}(e, \Delta) = \max_i \left[\sum_{i'} I_{c,i,c',i'}(e, \Delta) \right].$$

In Fig. 2, B1 and B2 each leak a total of $1+2=3$ bits, so the leakage from B to R under $\langle NR, C \rangle$ is 3 bits.

Finally for the $\langle R, C \rangle$ case, there are cumulative effects across both victim and adversary VMs

$$\langle R, C \rangle : I_{c,c'}(e, \Delta) = \sum_i \sum_{i'} I_{c,i,c',i'}(e, \Delta).$$

In Fig. 2, the sum of all bits leaked from B's VMs to R's VMs is 6, so the leakage from B to R under $\langle R, C \rangle$ is 6 bits.

Given the client-to-client information leakage in any one of the four models, the total information leakage that we seek to minimize is

$$I_{total}(e, \Delta) = \sum_c \sum_{c'} I_{c,c'}(e, \Delta).$$

Minimizing $I_{total}(e, \Delta)$ is equivalent to minimizing the average leakage across client pairs. We acknowledge that there may be other ways to represent the security of the system, for example, by minimizing the inter-client leakage. Although we chose the above representation for ease of comparison with prior work [13], the approaches we propose in Section 5 are general and can be used for such representations of security by modifying the objective function.

Problem Statement. We now pose the scheduling problem that is solved at the beginning of each epoch with the intent of minimizing information-leakage. We first choose and fix a particular information leakage model from amongst $\{R, NR\} \times \{C, NC\}$.

Inputs: The sets X, S, A, F , and an integer, m —see Table 2.

TABLE 2
The Inputs to the Scheduling Problem

Symbol	Meaning
X, x_i	$X = \{x_1, x_2, \dots, x_n\}$ is a set of integers where x_i denotes the number of VMs that belong to the i^{th} client.
S, s_j	$S = \{s_1, s_2, \dots, s_k\}$ is a set of integers where s_j denotes the number of VMs the j^{th} server can host.
A, a_i	$A = \{a_1, a_2, \dots, a_n\}$ is a set of integers where a_i denotes the number of VMs belonging to the i^{th} client which have arrived in this epoch.
F, f_l	$F = \{f_{l-1}, f_{l-2}, \dots, f_{l-\Delta}\}$, where each f_l encodes the placement of VMs in servers in the l^{th} epoch. Entries in F can be used to express VMs that depart.
m	An integer that is the migration budget. Each kind of migration (e.g., swap VMs on two servers) has a cost associated with it. A special symbol, ∞ , for m , is used to indicate that we are to be unconstrained by a migration budget.

Output: A placement for the current epoch that corresponds to the minimum total information leakage subject to the constraints specified as input.

As an example, suppose that the input is as shown for Epoch 2 in Fig. 1. Assume the leakage model is $\langle R, C \rangle$, and $m = 2$. Also assume that Client A corresponds to Client 1, Client B to Client 2 and so on. Then, the inputs are: $X = \{3, 1, 2\}$, $S = \{4, 4\}$, $A = \{0, 0, 0\}$ and a function $f_2(i, j)$ which indicates the number of VMs of client i on server j during Epoch 2, e.g., $f_2(1, 1) = 2$. The output is the placement shown for Epoch 3 in Fig. 1. In this example, the total information leakage in Epoch 2 is 10 bits, and in Epoch 3 it is 14 bits (10+4). That is, migrating some VMs causes the increase in information-leakage to be 4 only, rather than 10, which would have been the case had we adopted the same placement for Epoch 3 as we have for Epoch 2.

3 COMPUTATIONAL COMPLEXITY

We ask: does there exist an efficient algorithm for the problem from the previous section? If the problem is intractable (e.g., NP-hard), then the answer is, ‘no’, under customary assumptions (e.g., $P \neq NP$). In this section, we identify the computational complexity of the problem.

In Section 3.1 we present an upper bound on the complexity: a decision problem that is related polynomially is in NP. In Section 3.2 we present a lower-bound: the problem is NP-hard. In Section 3.3, we establish that the special case that is implemented in the version of Nomad we have been provided is NP-hard as well. In Section 3.4 we discuss ways in which the hardness can be mitigated, and some interesting problems that remain open.

3.1 An Upper Bound: NP

Consider the decision version of the secure scheduling problem in which we are given a threshold information leakage q , and we ask if there exists a placement for the next epoch such that all the input constraints are satisfied and the total information leakage incurred is at most q . This problem is in NP. An efficient (polynomially sized) certificate is a placement for the next epoch. We can verify the certificate against all the constraints in polynomial time: given a placement, we can use it together with the input F , which

is a set of placements used in prior epochs, to efficiently compute the information leakage it incurs, and compare this to the input threshold q . We can also efficiently check whether the placement respects server capacities and assigns each VM to exactly one server. Finally, we can compare this output placement to the placement of the immediately prior epoch to determine the number of migrations needed, and compare this to the budget m .

The optimization problem posed in Section 2 can be reduced in polynomial time to the above decision problem by doing a binary search on the threshold to find the minimum information leakage. Hence given that the decision version is in NP, an approach to solve the optimization problem is to use an oracle for the decision version. We discuss ILP and CNF-SAT solvers as oracles in Section 5.

3.2 A Lower Bound: NP-hard

To prove that the problem is NP-hard, we show that a problem which is known to be NP-hard can be reduced to it. We pose a variant of the classical Set Partition problem, which is known to be NP-hard [19]. We call this variant *Equal Cardinality Partition* and show that it is NP-hard as well. For the $\langle NR, NC \rangle$ model, we show an efficient reduction from Equal Cardinality Partition. For the other three models, we show an efficient reduction from Set Partition. For all four leakage models, we reduce to a sub-case of the decision version of our problem, that we call Initial Placement, thereby proving that the general problem is NP-hard.

We begin with Set Partition, which is known to be NP-complete [18]. In Set Partition, we are given a multiset $S = \{x_1, x_2, \dots, x_n\}$ of positive integers, and we seek to partition S into two subsets S_1 and S_2 such that the sum of the integers in S_1 equals the sum of the integers in S_2 . If indeed this is the case for a particular instance of the problem, we say that the instance is true and call the sets S_1, S_2 a solution to the instance. Otherwise we say that the instance is false.

In Equal Cardinality Partition, we impose the additional constraint that $|S_1| = |S_2|$. We also require that in every input instance, both the number of integers, n , as well as the sum of the integers, $\sum_i x_i$, are even, since otherwise the instance has no solution. We can check these two conditions efficiently and reject any input that does not meet them.

Lemma 3.1. *Equal Cardinality Partition is NP-hard.*

Our proof for Lemma 3.1 is in the appendix [50]. It is a reduction from Set Partition.

Now consider the following sub-case of the problem we pose in Section 2. We are given a set $X = \{x_1, \dots, x_n\}$ of client VMs, where $\sum_i x_i$ is even, and a set $S = \{s_1, s_2\}$ of two servers only, with $s_1 = s_2 = (\sum_i x_i)/2$. We can infer S from X ; that is, S does not have to be explicitly specified in the input for this sub-case of the general problem. The other inputs are $A = X$, $F = \emptyset$, and $m = \infty$. That is, we are asked to place the VMs from X across two equal-capacity servers in a way that minimizes information leakage. The capacity of each server is exactly half the total number of VMs that need to be placed. We show that this sub-case which we call the Initial Placement problem, is NP-hard, thereby proving that the general problem is as well.

The above version of the Initial Placement problem is an optimization problem. To show that it is NP-hard, we

consider the following decision version of it. It takes all the same inputs, i.e., X, A, F and m , and an additional input, q , an integer that is a threshold for the information leakage. The instance is true if there exists a placement which results in an information leakage of at most q bits, and false otherwise. The decision version can be reduced to the optimization version. Given an oracle for the optimization version, we simply query it with the inputs X, A, F and m , and compare its output to q . Thus, if this decision version is NP-hard, so is the optimization version.

Before we show that Initial Placement is NP-hard, we state Lemma 3.2 which is used in the reductions. The proof for Lemma 3.2 is in the appendix [50].

Lemma 3.2. *For any instance of Initial Placement, a lower-bound on the minimum information leakage for the $\langle NR, NC \rangle$ model is the one that arises when all the VMs of exactly half the clients are on one server, and all the VMs of the other half of the clients are on the other server. And this is the only placement that achieves this lower-bound. For the other three models, the lower-bound is achieved when each client has all of its VMs on the same server. And this is the only placement that achieves this lower-bound.*

Theorem 3.3. *Initial Placement is NP-hard.*

Proof. For the $\langle NR, NC \rangle$ model, the proof is a reduction from Equal Cardinality Partition. Given an instance, $S = \{x_1, x_2, \dots, x_n\}$, of Equal Cardinality Partition, let $X = \{x_1, x_2, \dots, x_n\}$, and $s_1 = s_2 = (\sum_i x_i)/2$. We set q to be the lower-bound on the minimum information leakage described in Lemma 3.2. That is, we set $q = 2 \times 2 \times \binom{n/2}{2}$. The “ $\binom{n/2}{2}$ ” term arises from having $n/2$ clients on each server. It gives us the number of pairs of clients on each server. One of the terms “2” in the expression arises from considering both pairs $\langle c, c' \rangle$ and $\langle c', c \rangle$ of clients. We do this as information leaks in both directions. The other “2” term arises from having two servers. This reduction is computable efficiently, in linear time. We assert that the instance of Equal Cardinality Partition is true if and only if the corresponding instance of Initial Placement is true.

Assume that the input Equal Cardinality Partition instance is true. Therefore, there exists a solution, two equal sized subsets, S_1 and S_2 , whose contents sum to the same. These two subsets give us the placement across the two servers which results in an information leakage of q for the resulting Initial Placement instance under the mapping. This follows directly from Lemma 3.2. Hence, the instance of Initial Placement produced by the mapping is true. For the other direction, assume that the instance of Initial Placement output by the mapping for some input instance of Equal Cardinality Partition is true. By Lemma 3.2, we know that this means there exists a placement such that all the VMs of exactly half of the clients are on one server, and all the VMs of the other half of the clients are on the other server. Then a solution to the Equal Cardinality instance is $S_1 = \{x_i \in X : \text{client } i \text{ is placed on server 1}\}$ and $S_2 = \{x_i \in X : \text{client } i \text{ is placed on server 2}\}$. By Lemma 3.2 the cardinalities of the two

sets are equal. Since each server has a capacity equalling half the total number of VMs, the contents of S_1 and S_2 sum to the same.

For the other three models, the proof is a reduction from the version of Set Partition in which the sum of all integers in S is even, to Initial Placement. This version of Set Partition is NP-hard—the proof is simply that if the sum is odd for an instance of Set Partition, that instance is trivially false. Given an instance, $S = \{x_1, x_2, \dots, x_n\}$, of Set Partition, let $X = \{x_1, x_2, \dots, x_n\}$, and $s_1 = s_2 = (\sum_i x_i)/2$. We set q to be the lower bound on the minimum information leakage described in Lemma 3.2. For the $\langle R, C \rangle$ model, this lower bound is $\sum_i (T/2 - x_i) \times x_i$, where T is the total number of VMs. The other two models are symmetric (see appendix [50]), and the lower bound for both is $\sum_i (T/2 - x_i)$. The reduction can be computed efficiently. We assert that the input Set Partition instance is true if and only if the resulting Initial Placement instance is true, using reasoning that is similar to the one used to demonstrate correctness for the reduction from Equal Cardinality Partition to Initial Placement in the $\langle NR, NC \rangle$ model. \square

The Initial Placement problem is a sub-problem of the general problem that is posed in Section 2. From Theorem 3.3, it follows that the general problem, i.e., for any $n \geq 2$ servers, is also NP-hard.

3.3 Closed Migration Under $\langle R, C \rangle$

We have shown in the previous section that the problem is NP-hard in general. In this section, we consider a particular special case of the problem: closed migration, under the $\langle R, C \rangle$ model for information-leakage. With closed-migration, there is a set of existing VMs, and no VMs arrive or leave. A reason we consider this special case in particular is that the implementation of Nomad we have been provided supports this special case only (see Section 6 under “Design Choices”). Consequently, for the approaches based on reduction to ILP and CNF-SAT that we discuss in Section 5 and have implemented to compare empirically with Nomad, this special case is of particular interest.

We observe that under the $\langle R, C \rangle$ model for information leakage, a placement, f , needs to express the number of a client’s VMs that are placed on a server only, and not the particular VMs that are placed on the server. That is, f can be perceived as a function, $f: [1, n] \times [1, k] \rightarrow [0, \max_i \{x_i\}]$, and can be encoded as a table of size $n \times k \times \log \max_i \{x_i\}$ bits. This is possible because the formula for inter-client information leakage does not contain any maximization (i.e., “max”) terms (see Section 2).

It is easy to compute a placement that expresses which specific client-VMs are placed on a server given such a more compact encoding for placement. Following is an algorithm. Let the new placement be f_t and the immediately prior placement be f_{t-1} . Let $x_{ij} = f_t(i, j) - f_{t-1}(i, j)$. Place all newly arriving VMs in a pool. For all $i \in [1, n]$ (i.e., the clients) and $j \in [1, k]$ (i.e., the servers), if $x_{ij} < 0$, remove $|x_{ij}|$ VMs of client i from server j and place them in the pool. Then, for all $i \in [1, n]$ and $j \in [1, k]$, if $x_{ij} > 0$, take x_{ij} VMs of client i from the pool and place them on server j . This algorithm guarantees the fewest migrations.

Computing such a placement after we have solved a problem-instance is more efficient: the input to and output from an algorithm for the optimization problem, and a corresponding decision problem, are more compact. We exploit this observation regarding encoding for the $\langle R, C \rangle$ model to assert the following theorem. Its proof is in the appendix [50].

Theorem 3.4. *Closed Migration under $\langle R, C \rangle$ is NP-hard.*

3.4 Mitigating NP-Hardness and Open Problems

In the appendix, we discuss ways we have explored to mitigate the NP-hardness of the scheduling problem, and the open problems that remain in this context.

4 ANALYSIS OF PRIOR APPROACHES

In this section we revisit and analyze the work of Moon et al. [13]. To our knowledge, that work is the first to propose the models of information leakage that we adopt. That work considers three approaches: a mapping to ILP, an initial design for a greedy algorithm called Baseline Greedy, and a final greedy algorithm called Nomad. We describe and address each, in turn, in Sections 4.1–4.3 below.

4.1 Mapping to ILP

The mapping to ILP of Moon et al. [13] comprises eight sets of constraints plus a constraint that expresses the optimization objective. The objective for a solver is to decide the values of several binary indicator variables that are used to encode placement. That is, there is a variable which takes value 1 if and only if a particular VM of a client resides on a particular server. An example of a constraint on those variables is that the sum of 1 values for those variables for a server, across all clients, should not exceed the capacity of the server. The solver is asked to minimize an expression for information leakage that uses those variables.

In assessing the mapping to ILP, we distinguish two properties: correctness and efficiency. We deem such a mapping to be correct if it is a reduction. That is, with the mindset that we map a decision version of the problem to a decision version of ILP, does the mapping have the property that the input instance is true if and only if the output instance is true? The mapping of Moon et al. [13] does appear to be a reduction. We deem the mapping to be efficient if there exists a polynomial-time algorithm that computes it. Unfortunately, the mapping is not.

Theorem 4.1. *Moon et al. [13]’s mapping to ILP may result in an output whose size is exponential in the size of the input, and an algorithm to compute the mapping may run in exponential-time.*

Proof. We revisit our discussion on encoding placement in Section 3.3. Under the $\langle R, C \rangle$ model, for example, as we discuss there, it suffices to record the number of VMs of each client on a server, rather than their identities. The choice of using decision variables for each VM in the ILP instance results in an exponential explosion. Instead, it may be possible to only have variables that encode the number of VMs. This is exactly something we leverage in our reductions (see Section 5). \square

Our assessment of their mapping to ILP is motivated by the observation there that a commercial ILP solver takes longer than a day for even modestly sized inputs. Our observations above, and empirical results from our reduction to ILP (see Section 6) suggest that this inefficiency in the mapping may be the cause.

4.2 Initial Design: Baseline Greedy

Moon et al. [13] use the observation that their mapping to ILP performs poorly in practice as motivation for the design of a greedy approach. In Baseline Greedy, in addition to all the inputs specified in the general problem, there is an additional input that enumerates the types of migration-moves allowed. A free-insert is when a single VM is moved from one server to another. A k -way swap of VMs $\langle v_a, v_b, v_c, \dots, v_k \rangle$ means that VM v_a takes the place of v_k , v_b takes the place of v_a , v_c takes the place of v_b , and so on. A k -way swap incurs k migrations. Based on the types of moves allowed and the migration budget, Baseline Greedy generates a list of possible moves and computes the reduction in leakage that results from each. The move that yields the greatest decrease in leakage is chosen and the current placement is changed to reflect this move. This happens iteratively until the migration budget runs out.

Theorem 4.2. *Baseline Greedy runs in time $\Theta(n!)$ in the worst-case. This is strictly worse than the $\Theta(2^n)$ upper-bound to determine an optimal placement.*

Proof. We first observe that given k VMs, there are $k!$ possible k -way swaps between them. The worst-case run-time for Baseline Greedy is when the set of allowed moves includes all possible n -way swaps, where n is the total number of VMs, and it is $\Theta(n!)$. This is strictly worse than $\Theta(2^n)$. Even if the set of allowed moves is only all k -way swaps, if k is $p(n)$, a polynomial in n , e.g., \sqrt{n} , the run-time is $\Theta(n^{p(n)})$ which also is strictly worse than $\Theta(2^n)$. So Baseline Greedy is worse than, for instance, reduction to CNF-SAT and then brute-force. We know that $\Theta(2^n)$ is an upper-bound because the decision version of the problem is in NP. \square

We acknowledge that Baseline Greedy is used only as a starting point, and its inefficiency is recognized by Moon et al. [13], albeit not as precisely as we do. However, a starting point that is strictly worse than the worst possible, suggests weaknesses in the design. Said more constructively, an identification of an upper-bound for computational complexity helps with the identification of a more appropriate starting point for the design.

4.3 Final Design: Nomad

Moon et al.’s final design is a greedy algorithm called Nomad [13]. Nomad is Baseline Greedy with the following modifications. (1) Instead of recomputing the benefit for each move after a move is selected, the benefit is only recomputed for the moves involving client pairs which were affected by the selected move. (2) Servers are grouped into clusters. Each client is assigned to a cluster and can only migrate within it. A move therefore only affects clients whose VMs reside in the same cluster in which the move was made. (3) The types of allowed moves are restricted to

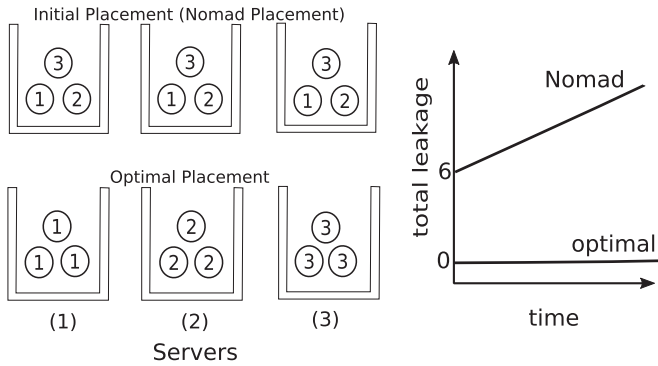


Fig. 3. An example of an instance for which Nomad performs no migrations, even though a placement incurring no leakage is possible. There are 3 servers, each having a capacity of at least 3 VMs, and 3 clients, each having 3 VMs. Under the $\langle NR, NC \rangle$, $\langle R, NC \rangle$ and $\langle NR, C \rangle$ models, no free-insert or pair-wise swap gives us a reduction in leakage, hence Nomad makes no moves. The optimal placement is shown at the bottom of the figure and incurs no leakage.

free-inserts and 2-way swaps only. (4) The entire move table is populated at the beginning of an epoch. Then, the move set is traversed starting from the move that gives the most benefit. If the claimed benefit from the time that benefit was computed lies within 95 percent of the current benefit and a move is feasible, then that move is made. If not, the move is re-inserted with an updated benefit.

In Nomad, as the moves are restricted to free-inserts and 2-way swaps, the running time is $\Theta(n^2)$ in the worst-case. Given that the problem is NP-hard (see Section 3.2), and the premise that $P \neq NP$, no polynomial-time algorithm exists which can give us the optimal value for every input. An important question we then ask is: what is the trade-off that is incurred by Nomad in exchange for its efficiency? We answer this question in Theorem 4.3 below. In the statement of the theorem, we refer to a class of inputs. We define it as follows. We observe that the number of servers, call it, k , is an input to the problem. Each value of $k \geq 2$, corresponds to a class of inputs. We point out that within each such class of inputs, there exist infinitely many inputs. This is because the capacity of each server is unbounded in any input.

Theorem 4.3. *$P \neq NP$ implies that there exist infinitely many classes of inputs, each with infinitely many inputs, for which a reduction in information leakage is possible, but for which Nomad provides no decrease in information leakage whatsoever, even when unconstrained by a migration budget.*

We emphasize that the above theorem says that Nomad does not even provide a decrease in information leakage, let alone minimize it. We emphasize also, that the theorem is true for all four leakage models.

Proving Theorem 4.3 The complete proof for Theorem 4.3 is in the appendix [50]. We provide an overview and intuition here. The overall intuition is that if the statement of Theorem 4.3 is false, then $P = NP$. We establish this contradiction in four stages. We first introduce a new, related problem, that we call the “Reduce Information Leakage for Two Servers” problem. This is the problem of determining, if possible, a new placement with lower information leakage for some input that has two servers only. We show that this problem is intractable, i.e., NP-hard, under polynomial-time

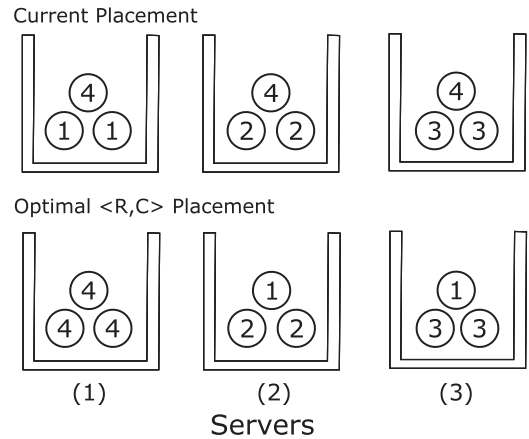


Fig. 4. An example to show the sub-optimality of Nomad for the $\langle R, C \rangle$ model. The capacity of each server is at least 3 VMs. Clients 1, 2 and 3 have two VMs each, and Client 4 has three VMs. The placement at the top of the figure is the current placement. There is no free-insert or 2-way swap that results in a reduction in leakage. Hence Nomad makes no moves even when unconstrained by a migration budget. The optimal placement is shown in the bottom of the figure.

Turing reductions (Lemma 8.1 in the appendix [50]). Then, we show that there must exist at least one two-server input for which a reduction in information leakage is possible, but Nomad makes no moves (Lemma 8.2 in the appendix [50]). We then show that, if so, there must exist infinitely many such inputs (Lemma 8.3 in the appendix [50]). We then show that for every unique two-server input for which Nomad provides no reduction in information leakage, there exists a unique n -server input, for every $n \geq 2$, for which Nomad provides no reduction in information leakage (Lemma 8.4 in the appendix [50]). We finally tie all these intermediate results together to prove Theorem 4.3.

Examples. As a concrete example of a class of inputs for which Nomad provides no benefit, consider the following for the $\langle NR, NC \rangle$, $\langle R, NC \rangle$ and $\langle NR, C \rangle$ models. We have n servers, each of capacity of at least n , and n clients each with n VMs, where $n \geq 3$. In the initial placement, each server has a VM that belongs to each client. Fig. 3 shows the current placement when $n = 3$. The optimal leakage is 0, obtained from the placement in which all the VMs of client i are on server i . However on running Nomad on this instance, with an unbounded migration budget, we find that no free-insert or 2-way swap yields a reduction in leakage. So, Nomad does not change the current placement, which is sub-optimal. Indeed, for the $\langle NR, NC \rangle$ model, the current placement incurs the worst possible information leakage.

As another example, for the $\langle R, C \rangle$ model, consider the following class of inputs. There are n servers, each with capacity of at least n , and $n + 1$ clients. Clients $1, 2, \dots, n$ have $n - 1$ VMs each, and client $n + 1$ has n VMs. In the current placement, server i has the VMs: $\langle i, 1 \rangle, \langle i, 2 \rangle, \dots, \langle i, n - 1 \rangle, \langle n + 1, i \rangle$, where $\langle x, y \rangle$ corresponds to VM y of client x . The current placement and optimal placement for $n = 3$ are shown in Fig. 4. There is no free-insert or 2-way swap which yields a reduction in leakage. Hence Nomad makes no changes to the current placement. We have tried this input against the Nomad software which we have been provided, and confirmed the above observation.

Algorithm 1. Reduction to ILP for the $\langle R, C \rangle$ Model for Information Leakage. C, Ob are the Outputs, $|\cdot|$ is Absolute Value, and the Other Symbols are Explained in Table 2 in Section 2.

Line (1) Ensures that the Migration Budget, m , is Respected. Lines (2)–(3) Ensure that the Capacity of a Server, s_j , is not Exceeded. Lines (4)–(5) Ensure that Every VM of Each Client is Placed on Exactly One Server. The Objective Function on Line (6) Minimizes the Total Information Leakage

Input: An instance of the scheduling problem, $\langle X, S, A, m, F \rangle$

Output: An ILP instance: (i) a set of linear constraints C , and, (ii) an objective function Ob

$$1: C \leftarrow \sum_{i \in [1, n]} \sum_{j \in [1, k]} \|f_t(i, j) - |f_{t-1}(i, j)|\| \leq 2 \times m;$$

2: **foreach** $s_j \in S, j \leftarrow 1$ **to** k **do**

$$3: C \leftarrow C \cup \sum_{i \in [1, n]} |f_t(i, j)| \leq s_j;$$

4: **foreach** $i \leftarrow 1$ **to** n **do**

$$5: C \leftarrow C \cup \sum_{j \in [1, k]} |f_t(i, j)| = x_i;$$

$$6: Ob \leftarrow \min \sum_{\substack{(c, c') \\ c, c' \in [1, m] \\ c \neq c'}} \sum_{t' \in [t-\Delta, t]} \sum_{j \in [1, k]} f_{t'}(c, j) \times f_{t'}(c', j);$$

7: **return** $\langle C, Ob \rangle$;

5 REDUCTIONS TO ILP AND CNF-SAT

Given that Nomad provides no decrease in information leakage for infinitely many inputs, we revert to the two approaches that we discuss in this section. One is an efficient reduction to ILP and the other is an efficient reduction to CNF-SAT. Our primary intent is to identify whether we can indeed realize software that is efficient in practice, and provides the security guarantee of minimizing information leakage, notwithstanding the NP-hard worst-case for computational complexity. Both approaches are efficient algorithms given access to oracles for ILP and CNF-SAT, respectively. We chose to invest in both approaches because there can be differences in the performance of individual solvers for different classes of problem instances.

Reduction to ILP. Algorithm 1 is our reduction to ILP for the $\langle R, C \rangle$ case. The other cases require minor modifications only, as we discuss in the following. The input to the reduction is an instance of the problem posed in Section 2, and the output is the corresponding ILP instance. The unknown in the constraints and the optimization objective, is a new placement—values for the $n \times k$ variables $f_t(i, j)$, for each $i \in [1, n], j \in [1, k]$, for a given t , for the $\langle R, C \rangle$ case. These values are the number of VMs of client i on server j in the upcoming epoch, t . For the other three models of information leakage, the unknowns are $f_t(v, i, j)$, where $v \in [1, x_i]$, which indicates whether VM v of client i is placed on server j in epoch t .

In Algorithm 1, Line (1) initializes the output set of constraints C , with a constraint that ensures that the migration budget, m , is not exceeded by whatever moves are needed to reach the new placement, f_t . If m is specified as ∞ in the input, we initialize $C \leftarrow \emptyset$ instead. Lines (2)–(3) add constraints to ensure that the capacity of each server, s_j , is not

exceeded in the new placement. Lines (4)–(5) add constraints to ensure that every VM of each client is placed on exactly one server. The objective function, Ob , that is instantiated in Line (6), ensures that we minimize the total information leakage with the new placement. The only nuance here is that the objective function, Ob , for the $\langle R, C \rangle$ case, is not a set of linear constraints, but rather quadratic constraints. However, Quadratic Constraint Programming is known to be in NP as well [20]—we have designed and implemented a reduction from quadratic to linear constraints via bit-wise multiplication.

Algorithm 1 runs in polynomial-time; its running-time is: $O\left(\binom{n}{2} \cdot \Delta \cdot k \cdot \log_2^2 \max_i x_i\right)$, where n is the number of clients, Δ is the number of epochs in the sliding-window, k is the number of servers, and $\log_2 \max_i x_i$ is the maximum number of bits we need to express the number of VMs per client. The reason is that the running time is dominated by Line (6)—the above expression for the running-time comes directly from the three summations and the product. This bound is not tight; more efficient reductions can exist from, for example, more compact encodings for placement.

Reduction to CNF-SAT. Our reduction to CNF-SAT is first to satisfiability for boolean circuits, CIRCUIT-SAT, and then a well-known linear-time reduction from CIRCUIT-SAT to CNF-SAT [17]. The algorithm and discussion are in the appendix [50]. As we discuss there, the reduction to CIRCUIT-SAT encodes the same constraints as the above reduction to ILP, and the running-time of the reduction is the same.

Comparison with the approaches of prior work. We can make the following two comparisons between our reductions and the work of Moon et al. [13].

- Comparison with the reduction to ILP in that work: as we state in Theorem 4.1, the reduction there can be exponential-time in the worst-case. The proof for Theorem 4.1 refers to Section 3.3, in which we establish the existence of a more compact encoding for placement for the $\langle R, C \rangle$ model. When this compact encoding is used for the input, the reduction there is exponential-time. In contrast, both our reductions are polynomial-time. We attribute this crucial difference in the reductions to our observations in Section 6.2 for Test 1.
- Comparison with Nomad: Nomad runs in polynomial- (quadratic-) time. However, it cannot be called an algorithm for the problem we, and Moon et al. [13], consider, because, as Theorem 4.3 in Section 4.3 states, it simply does not work at all for lots of inputs. Our reductions, coupled with an oracle for ILP or CNF-SAT, which is what the corresponding solvers are intended to be, are correct algorithms.

6 EMPIRICAL ASSESSMENT

We have conducted a limited empirical assessment of the two approaches from the previous section, and Nomad. The intent of our empirical assessment is: (a) to understand the overhead of actually providing a security guarantee of minimizing leakage in practice, and, (b) validate empirically our analytical insights on the shortcomings of Nomad.

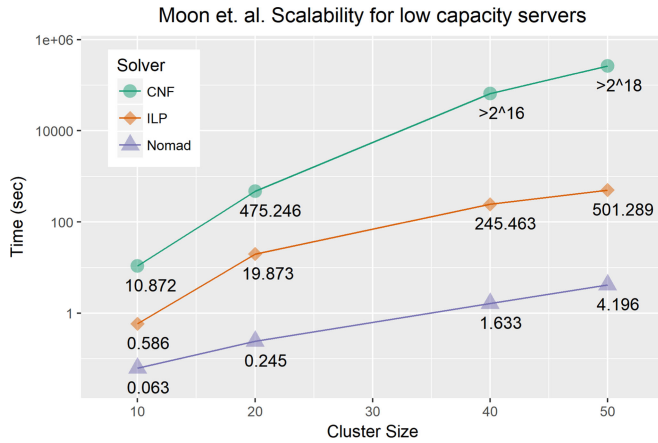


Fig. 5. Scalability of Nomad, ILP, and CNF-SAT using the testing parameters presented in Moon et. al [13]. Cluster size of x means: x servers with capacity 4, and x clients with 2 VMs each. Total VMs in the cluster: $\sum VM = 2 \cdot x$.

6.1 Tests

We designed and ran the following tests.

Test 1. A test for scalability. We report the response-time versus cluster-sizes for the two approaches from the previous section, and Nomad.

Test 2. Demonstrate an example for which Nomad provides no decrease in information-leakage. We know that several such inputs exist—see Theorem 4.3 in Section 4.3. We demonstrate one, in practice.

Test 3. Random input-cases that have a known, non-zero value for optimum information-leakage. The intent is to check whether each of the three approaches indeed converges to the optimum. This test is different from Test 2 in that in Test 2, we pick an input that we have determined analytically beforehand is one for which Nomad provides no benefit. In Test 3, on the other hand, we do not have such *a priori* knowledge.

Test 4. A test for how well an approach does given a migration budget. We test with random inputs, with three migration budgets, and measure the incremental improvements to the information leakage.

Test 5 Another test for scalability. Number of clients, servers and VM-slots per server, versus response-time for 1 epoch. We choose the inputs such that a client has several VMs, and server-utilization is 50 percent only. We anticipate that this models reality somewhat closely—in practice, cloud-servers are under-utilized [21].

Setup. All tests were performed on a desktop with an Intel (R) Core(TM) i7-4790 CPU that clocks at 3.60 GHz, has 24 GB RAM, and runs the 64-bit Windows 8.1 operating system. The code for the reductions is written in Java. Our ILP-solver was CPLEX, commercial version 12.6.1.0 [22], the same as the one used in prior work [13]. CPLEX provides a Java API, which we used. Our SAT-solver was Lingeling [23]. Lingeling does not have native binaries for Windows; we used Cygwin [24]. We modified Nomad to allow for arbitrary inputs, and for retrieval of the placement at each epoch. The observations for Nomad were not impacted by these modifications.

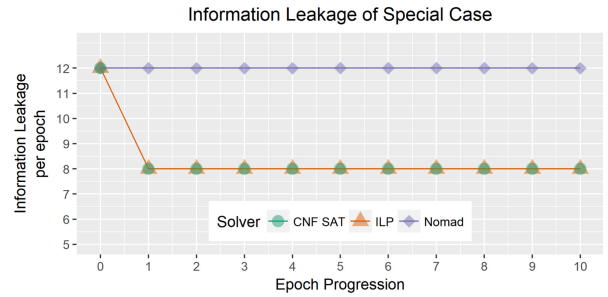


Fig. 6. This figure shows how information leakage varies when all three approaches are allowed to run 10 epochs on the placement map shown in Fig. 4. The CNF-SAT and ILP solvers were both able to obtain the optimal placement in which a leakage of 8 bits is incurred per epoch. Nomad was unable to make any move.

Design Choices. We made the following design choices:

- 1) We ran tests for the $\langle R, C \rangle$ leakage model only. The reason is that the implementation of Nomad we were provided works for our inputs, under $\langle R, C \rangle$ only.
- 2) We ran tests for the closed migration version of the problem only (see Section 3.2). The reason is that the implementation of Nomad we were provided works for closed-migration only.
- 3) The sliding window for was fixed at 5 epochs (see Section 2). The reason is that the implementation of Nomad we were provided has the sliding window hard-coded at 5.

6.2 Results

We now present and discuss our results.

Test 1. We show one of our results in Fig. 5, for a test that we reprised from the work of Moon et al. [13]. The input instances comprised $x \in \{10, 20, \dots, 50\}$ servers each with a capacity of 4 VMs, and x clients each with 2 VMs. The initial placement is random. The optimal placement is for every client to be placed on a server of its own. Fig. 5 shows the average time to compute the placement per epoch. We find that Nomad performs well, as reported in prior work [13]. Nomad’s response-time is not as good in some of our other tests, however; see below.

An interesting observation is a difference in our results, and the results reported by Moon et al. [13], on the performance of the approach based on reduction to ILP. Moon et al. [13] report that their reduction to ILP takes longer than a day for cluster sizes of 40 and 50. In contrast, as we report in Fig. 5, our ILP approach takes about 4 and 8 minutes, respectively. We suspect that this is a consequence of their inefficient (exponential-time) reduction to ILP.

Test 2. We show the results in Fig. 6. In this test, we ran the example from Fig. 4 in Section 4. We establish analytically in that section that Nomad provides no decrease in information-leakage for such an input, even with an unlimited migration budget. Our empirical observations concur. The approaches based on reduction to CNF-SAT and ILP immediately converge to the optimum.

Test 3. We show the results for this test in Fig. 7. Our input instance comprised 5 servers, each with a capacity of 4 VMs, and 6 clients, each with 3 VMs. We ran the test over 1 epoch for 10 different random initial placements. The

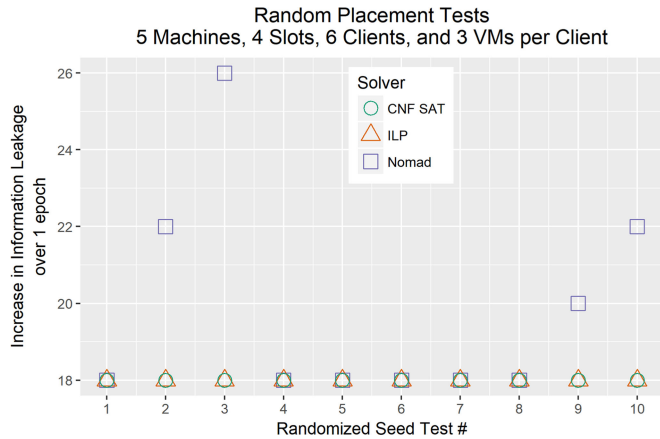


Fig. 7. Increase in information leakage for 10 random initial placements for input instances of 5 servers with 4 VM slots each, and 6 clients with 3 VMs each. Each of the 10 tests were run over a duration of 1 epoch with an infinite migration budget. The optimal placement is the one resulting in leakage of 18 bits. ILP and CNF-SAT always converged to the optimal. Nomad’s performance was inconsistent.

optimal placement results in an increase in total information leakage of 18 bits. We see that CNF-SAT and ILP were always able to converge to an optimal placement, whereas Nomad yields the optimum sometimes only.

Test 4. The results are shown in Fig. 8. The input instance comprised 6 servers each with a capacity of 6 VMs, and 3 clients each having 6 VMs. We used a random initial placement, and provided three different migration budgets: 2, 5, and ∞ . The results in Fig. 8 show that the ILP and CNF-SAT solvers reached the optimal placement faster than Nomad. This is because Nomad makes locally optimal greedy choices; the other two approaches globally optimize for information leakage.

Test 5. Our results are shown in Fig. 9, our input instance comprised x servers, where x ranged from 5 to 20. Each server had a capacity of x VMs. We had x clients each with $\lfloor x/2 \rfloor$ VMs; that is, the number of total VMs ranged from $5 * \lfloor 5/2 \rfloor = 10$ to 200. We adopted a time-out of 4 minutes per test. For the initial placement, we tried (i) random—VM assigned randomly to a server that can accommodate it, and, (ii) spread—each VM of a client is placed on a different server. That is, under the spread-placement, each client had its $\lfloor x/2 \rfloor$ VMs spread out on the first $\lfloor x/2 \rfloor$ servers; the other servers are empty. Both

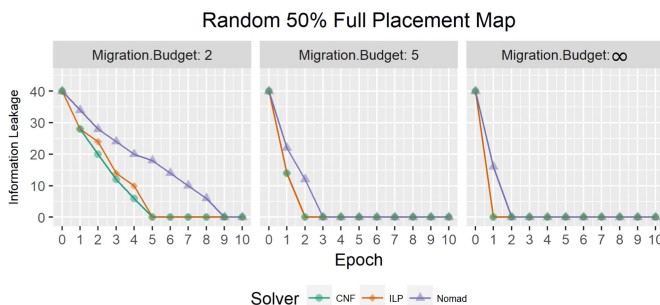


Fig. 8. How the per-epoch information leakage varied over 10 epochs for all three approaches with a random initial placement. The instance comprised 6 servers each having a capacity of 6 VMs, and 3 clients each having 6 VMs.

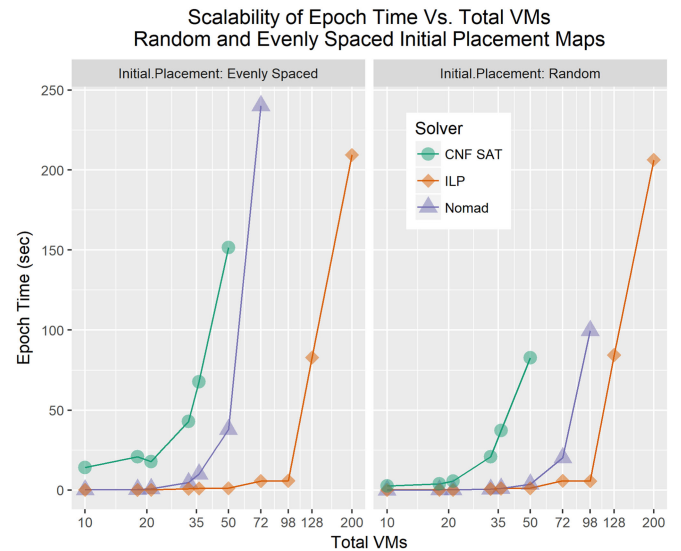


Fig. 9. The average time it took to compute a placement for 1 epoch. The input instance comprised of x servers each having a capacity of x VMs, and x clients each having $\lfloor x/2 \rfloor$ VMs. The test was run for both random and even initial placements. Tests were not included if they did not complete in under 240 seconds (4 minutes).

random- and spread-placements are used in practice [25]. Fig. 9 shows that the ILP implementation scaled the best. This test suggests contrasts with Test 1 above, in that it demonstrates that an approach based on reduction to ILP can perform well; in some cases, even better than a polynomial-time algorithm such as Nomad.

6.3 Takeaways

From our empirical results, we see that our approaches that provide security incur a higher overhead than Nomad. However, this overhead, in practice, is not as high as suggested by prior work. A main thrust of our work is to identify what the overhead is. We find that if an efficient reduction to ILP is devised and adopted, then the run-time is significantly lower than previously thought. Whereas we are able to guarantee minimum information-leakage, it is unclear what security benefit more efficient approaches, particularly Nomad, actually provide.

The main takeaway is: if we are to propose VM-migration as a viable technique to reduce information-leakage at scale, then more research-investment is needed, e.g., into more efficient reductions, and solvers, before it can be deemed to be practical. Our results suggest that such a technique can be practical for relatively small clouds only—ones with 10’s, but not 100’s or 1000’s, of servers. Approaches such as Nomad suggest that this technical challenge can be bypassed easily. But this is simply not true. The reasons for this lie at the very foundations of computing.

7 RELATED WORK

We discuss work on cross-VM side-channels and defences against them.

Cross-VM side-channel attacks. Ristenpart et al. [5] were among the first to show how an attacker can map the internal cloud infrastructure using network probing in order to achieve co-residency with a victim VM, and how the co-

residency can be exploited to extract coarse-grained information such as keystroke patterns.

This was followed up by the work of Zhang et al. [8] who showed how a Prime+Probe side-channel attack on upper level cache can be used to extract more fine-grained information such as an ElGamal decryption key. In the prime stage, the attacker fills a portion of cache and then waits to allow the victim to access cache. In the probe stage, the attacker reloads the primed data and uses the probe time to decide whether the victim accessed that portion of cache. Prime+Probe attacks on last level caches that make use of huge page-sizes have also been studied [1], [3], [26].

Research has also shown how fine-grain information can be extracted using Flush+Reload techniques [2], [7], [9]. In the flush stage, the attacker flushes out certain lines in memory and waits to allow the victim to access them. In the reload stage, the attacker accesses the flushed lines. If any such line does not take long to load, the attacker knows the line was accessed by the victim.

Other works have shown that sharing of same-content memory pages among co-resident clients poses the threat of a memory disclosure attack [4], [6]. The attacker can identify which pages it shares with a victim and take advantage of differences in write times to gain information about the victim's applications and OS.

Of course, other kinds of side-channels may exist. For example, the work of Xu et al. [27] considers the issue of excluding the operating system from the trusted computing base, and side-channels that can result as a consequence. As another example, the work of Shinde et al. [28] considers side-channels based on page-faults. Such attacks can certainly underlie cross-VM side-channel attacks. For example, if a benign VM runs a particular kind of application that allows a malicious VM that is co-resident to cause a page-fault, it may be vulnerable to the latter kind of attack.

Defences. To counter such attacks, several defence strategies have been proposed. At a hypervisor level, Vattikonda et al. [10] proposed hiding a program's execution time by eliminating fine grained timers. Kim et al. [29] used statistical multiplexing to protect against cache-based side-channel attacks in the cloud. They proposed a set of locked cache lines per core that are efficiently multiplexed amongst co-resident VMs and never evicted from the cache. Raj et al. [11] identified last level cache (LLC) sharing as one of the impediments to finer grain isolation, and proposed two resource management approaches to provide performance and security isolation - cache hierarchy aware core assignment and page colouring based cache partitioning. Shi et al. [30] proposed an approach that leverages dynamic cache colouring: when an application is doing security-sensitive operations, the VMM is notified to swap the associated data to a safe and isolated cache line. Varadarajan et al. [31] suggested introducing a minimum run time (MRT) guarantee into the Xen scheduler, to limit the frequency of preemptions.

At a guest OS level, Zhang et al. [32] proposed a method by which a tenant can construct its VMs to automatically inject additional noise into the timings that an attacker might observe from caches. Pattuk et al. [12] suggested partitioning cryptographic keys into random shares and storing each share in a different VM. Zhou et al. [33] presented a

software approach which dynamically manages physical memory pages shared between security domains to disable sharing of LLC lines.

At a hardware level, some works have proposed modifying the cache architecture to obtain access randomization [34], [35] while others have explored the partitioning of cache [36]. In recent work, Liu et al. [37] demonstrated how to combat LLC side-channel attacks by using the Intel Cache Allocation Technology (CAT) to partition the LLC into a hybrid hardware-software managed cache.

As many side-channel attacks are based on the observances of memory access, a promising defence strategy is to use oblivious RAM (ORAM) to randomize data access patterns [38], [39]. When an adversary observes the physical storage locations accessed, the ORAM algorithm ensures that they have negligible probability of learning the true access pattern.

An approach that focuses on isolation is that of shielded execution, for example, by using Intel SGX enclaves [15]. Chen et al. [40] presented a software framework that enables a shielded execution to detect page-fault side-channel attacks.

Various scheduler-based defences have also been explored. Y. Zhang et al. [41] proposed periodically migrating VMs using methods based on game theory. Azar et al. [42] focused on mitigating co-location attacks, on the premise that they are a necessary first step to performing cross-VM attacks. They suggested assigning VMs to servers such that attack VMs are rarely co-located with target VMs. Their model is based on co-location rather than on information leakage. Han et al. [43] also studied how to minimize the attacker's possibility of co-locating their VMs with targets. They introduced a security game model to compare different VM allocation policies and advocated selecting an allocation policy statistically from a pool instead of having a fixed one. Bijon et al. [44] proposed an attribute-based constraints specification framework that enables clients to express essential properties of their resources as attributes. A constraints enforcement engine then schedules the VMs while respecting the conflicts specified by these attributes. Han et al. [45] proposed metrics which can be used to assess the security of a VM allocation policy.

The work that is most closely related to ours is by Moon et al. [13]. They characterized information leakage models which can be used to pose the security-sensitive scheduling problem precisely and explore various approaches to address the problem. Our work differs from theirs in the following ways: (1) We analyze the problem's computational complexity, (2) Our reduction to ILP is polynomial time, while theirs can be exponential time for a certain encoding of input, (3) We also investigate the approach of reducing to CNF-SAT and employing a SAT solver, and (4) Our approaches guarantee security whereas their final approach, Nomad, does not.

8 CONCLUSIONS

We have addressed problems related to the use of scheduling as an approach to mitigating cross-VM side-channel attacks in cloud systems. Such attacks can cause information to leak. Specifically, we have sought to analyze the overhead incurred by an algorithm that seeks to minimize information leakage, for models of information leakage that have been proposed in prior work [13]. We have posed and

answered a fundamental question: is the problem of determining the minimum information leakage tractable? We have established that it is intractable (**NP-hard**) by showing that even seemingly simpler sub-cases such as placing VMs across only two equal-capacity servers, and migrating VMs across only two equal-capacity servers, is **NP-hard**. Given the customary assumption, $\mathbf{P} \neq \mathbf{NP}$, this tells us that no efficient algorithm exists for the problem. We have identified, however, that a decision version to which the optimization problem relates to polynomially is in **NP**.

We have then analyzed prior work, which proposes an efficient, greedy approach called Nomad, motivated by an observation there that an approach based on reduction to ILP does not scale to even modestly-sized inputs that can arise in practice. We have identified that the reduction to ILP there can be inefficient, i.e., exponential-time. Also, we have identified deficiencies with Nomad: its initial design is unnecessarily inefficient, and the final design yields no decrease in information leakage whatsoever for at least infinitely many inputs, unless $\mathbf{P} = \mathbf{NP}$.

We have then designed and implemented efficient reductions to ILP and CNF-SAT, with the intent of adopting corresponding solvers as oracles. We have conducted a limited empirical assessment with a focus on the main points of our work. Our empirical results suggest that the overhead imposed by an approach that indeed decreases information leakage is more than prior work suggests, but that the news is not all bad. For inputs that prior work suggests take longer than a day with an ILP approach, our implementation that is based on an efficient reduction takes a few minutes only. This suggests that further research-investment in well-founded approaches can decrease the overhead further, for instances that are likely to arise in practice.

Apart from such a research-investment as future work, we have pointed to several open problems. Is the closed migration problem tractable under models other than $\langle R, C \rangle$? Is the general problem only weakly **NP-hard**? Is the optimization version of the problem efficiently approximable?

ACKNOWLEDGMENTS

We thank the authors of Nomad [13] for making a version of their implementation available to us.

REFERENCES

- [1] G. Irazoqui, T. Eisenbarth, and B. Sunar, "A shared cache attack that works across cores and defies VM sandboxing—and its application to AES," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 591–604.
- [2] G. Irazoqui, M. S. Inci, T. Eisenbarth, and B. Sunar, "Wait a minute! a fast, cross-VM attack on AES," in *Research in Attacks, Intrusions and Defenses*. Berlin, Germany: Springer International Publishing, 2014, pp. 299–319.
- [3] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *Proc. 36th IEEE Symp. Security Privacy*, May 2015, pp. 605–622.
- [4] R. Owens and W. Wang, "Non-interactive os fingerprinting through memory de-duplication technique in virtual machines," in *Proc. 30th IEEE Int. Perform. Comput. Commun. Conf.*, Nov. 2011, pp. 1–8.
- [5] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009, pp. 199–212.
- [6] K. Suzaki, K. Iijima, T. Yagi, and C. Artho, "Memory deduplication as a threat to the guest os," in *Proc. 4th Eur. Workshop Syst. Security*, 2011, pp. 1:1–1:6.
- [7] Y. Yarom and K. Falkner, "Flush+reload: A high resolution, low noise, L3 cache side-channel attack," in *Proc. 23rd USENIX Security Symposium*. San Diego, CA, USA: USENIX Association, Aug. 2014, pp. 719–732. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom>
- [8] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 305–316.
- [9] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-tenant side-channel attacks in PaaS clouds," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 990–1003.
- [10] B. C. Vattikonda, S. Das, and H. Shacham, "Eliminating fine grained timers in xen," in *Proc. 3rd ACM Workshop Cloud Comput. Security Workshop*, 2011, pp. 41–46.
- [11] H. Raj, R. Nathuji, A. Singh, and P. England, "Resource management for isolation enhanced cloud services," in *Proc. ACM Workshop Cloud Comput. Security*, 2009, pp. 77–84.
- [12] E. Pattuk, M. Kantarcioglu, Z. Lin, and H. Ulusoy, "Preventing cryptographic key leakage in cloud virtual machines," in *Proc. 23rd USENIX Security Symp.*, Aug. 2014, pp. 703–718. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pattuk>
- [13] S.-J. Moon, V. Sekar, and M. K. Reiter, "Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security*, 2015, pp. 1595–1606.
- [14] D. Evans, A. Nguyen-Tuong, and J. Knight, "Effectiveness of moving target defenses," in *Moving Target Defenses*. Berlin, Germany: Springer, 2011, pp. 29–48.
- [15] A. Baumann, M. Peinado, and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," in *ACM Transactions on Computer Systems*. New York, NY, USA: ACM, 2015, pp. 8:1–8:26.
- [16] M. Sindelar, R. K. Sitaraman, and P. Shenoy, "Sharing-aware algorithms for virtual machine colocation," in *Proc. 23rd Annu. ACM Symp. Parallelism Algorithms Archit.*, 2011, pp. 367–378.
- [17] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. New York, NY, USA: McGraw-Hill Higher Education, 2001.
- [18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.
- [19] R. M. Karp, *Reducibility among Combinatorial Problems*. Boston, MA, USA: Springer US, 1972, pp. 85–103. [Online]. Available: http://dx.doi.org/10.1007/978-1-4684-2001-2_9
- [20] S. A. Vavasis, "Quadratic programming is in NP," *Inf. Process. Lett.*, vol. 36, no. 2, pp. 73–77, Oct. 1990.
- [21] P. Nelson, "How one startup hopes to solve server underutilization," *Netw. World*, Aug. 2015. [Online]. Available: <http://www.networkworld.com/article/2959532/data-center/startup-says-it-has-solved-server-underutilization.html>
- [22] Cplex. [Online]. Available: <http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>, Accessed on: Nov. 2017.
- [23] Lingeling. [Online]. Available: <http://fmv.jku.at/lingeling/>, Accessed on: Nov. 2017.
- [24] Cygwin. [Online]. Available: <https://www.cygwin.com/>, Accessed on: Nov. 2017.
- [25] Docker swarm strategies. [Online]. Available: <https://docs.docker.com/swarm/scheduler/strategy/>, Accessed on: Oct. 2016
- [26] M. S. Inci, B. Gülmezoglu, G. I. Apechechea, T. Eisenbarth, and B. Sunar, "Seriously, get off my cloud! cross-VM RSA key recovery in a public cloud," *IACR Cryptology ePrint Archive*, vol. 2015, Art. no. 898.
- [27] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 640–656.
- [28] S. Shinde, Z. Chua, V. Narayanan, and P. Saxena, "Preventing your faults from telling your secrets," in *Proc. 11th ACM Symp. Inform. Comput. Commun. Security*, 2016, pp. 317–328.
- [29] T. Kim, M. Peinado, and G. Mainar-Ruiz, "Stealthmem: System-level protection against cache-based side channel attacks in the cloud," in *Proc. 21st USENIX Conf. Security Symp.*, 2012, p. 11.
- [30] J. Shi, X. Song, H. Chen, and B. Zang, "Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring," in *Proc. 2011 IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. Workshops*, 2011, pp. 194–199.

- [31] V. Varadarajan, T. Ristenpart, and M. Swift, "Scheduler-based defenses against cross-VM side-channels," in *Proc. 23rd USENIX Security Symp.*, 2014, pp. 687–702.
- [32] Y. Zhang and M. K. Reiter, "Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 827–838.
- [33] Z. Zhou, M. K. Reiter, and Y. Zhang, "A software approach to defeating side channels in last-level caches," *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 871–882.
- [34] F. Liu and R. B. Lee, "Random fill cache architecture," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2014, pp. 203–215.
- [35] Z. Wang and R. B. Lee, "A novel cache architecture with enhanced performance and security," in *Proc. 41st Annu. IEEE/ACM Int. Symp. Microarchit.*, 2008, pp. 83–93.
- [36] D. Page, "Partitioned cache architecture as a side-channel defence mechanism," 2005, page@cs.bris.ac.uk 13017 received 22 Aug. 2005. [Online]. Available: <http://eprint.iacr.org/2005/280>
- [37] F. Liu, et al., "Catalyst: Defeating last-level cache side channel attacks in cloud computing," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, Mar. 2016, pp. 406–418.
- [38] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, May 1996. [Online]. Available: <http://doi.acm.org/10.1145/233551.233553>
- [39] E. Stefanov, et al., "Path ORAM: An extremely simple oblivious RAM protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 299–310. [Online]. Available: <http://doi.acm.org/10.1145/2508859.2516660>
- [40] S. Chen, X. Zhang, M. K. Reiter, and Y. Zhang, "Detecting privileged side-channel attacks in shielded execution with déjà VU," in *Proc. ACM Asia Conf. Comput. Commun. Security*, 2017, pp. 7–18. [Online]. Available: <http://doi.acm.org/10.1145/3052973.3053007>
- [41] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, "Incentive compatible moving target defense against VM-colocation attacks in clouds," in *Gritzalis, Dimitris and Furnell, Steven and Theoharidou, Marianthi (eds.) Information Security and Privacy Research*. Berlin, Heidelberg, Germany: Springer, 2012, pp. 388–399.
- [42] Y. Azar, S. Kamara, I. Menache, M. Raykova, and B. Shepard, "Co-location-resistant clouds," in *Proc. 6th Edition ACM Workshop Cloud Comput. Security*, 2014, pp. 9–20.
- [43] Y. Han, T. Alpcan, J. Chan, and C. Leckie, "Security games for virtual machine allocation in cloud computing," in *Proc. 4th Int. Conf. Decision Game Theory Security - Vol. 8252*, 2013, pp. 99–118.
- [44] K. Bijon, R. Krishnan, and R. Sandhu, "Mitigating multi-tenancy risks in IaaS cloud through constraints-driven virtual resource scheduling," in *Proc. 20th ACM Symp. Access Control Models Technol.*, 2015, pp. 63–74.
- [45] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Using virtual machine allocation policies to defend against co-resident attacks in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 1, pp. 95–108, Jan./Feb. 2017.
- [46] N. Karmarkar and R. M. Karp, "An efficient approximation scheme for the one-dimensional bin-packing problem," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, 1982, pp. 312–320.
- [47] O. Goldreich, *Computational Complexity: A Conceptual Perspective*. New York, NY, USA: Cambridge University Press, 2008.
- [48] C. Sinz, *Towards an Optimal CNF Encoding of Boolean Cardinality Constraints*. Berlin, Heidelberg, Germany: Springer, 2005, pp. 827–831. [Online]. Available: http://dx.doi.org/10.1007/11564751_73
- [49] N. Mousavi, "Algorithmic problems in access control," 2014. [Online]. Available: <http://hdl.handle.net/10012/8303>
- [50] N. Juma, J. Shahan, K. Bijon, M. Tripunitara, "The Overhead from Combating Side-Channels in Cloud Systems using VM-Scheduling—Appendix," (2018). [Online]. Available: <https://ece.uwaterloo.ca/~tripunit/VMScheduling-overhead-appendix.pdf>

Nahid Juma received the MASc degree in electrical engineering from the University of Waterloo, in 2015. She is currently working toward the PhD degree with the University of Waterloo. She researches information security, applied cryptography and cloud computing.

Jonathan Shahan received the MASc and BAsC degrees in computer engineering from the University of Waterloo, in Canada, where he is currently working towards the PhD degree. He researches information security and machine learning.

Khalid Bijon received the PhD degree in computer science from the University of Texas at San Antonio. He is a member of technical staff at Composure.ai. He researches access control, secure information sharing, and its applications to cloud computing.

Mahesh Tripunitara is an associate professor in the Electrical and Computer Engineering (ECE) Department, the University of Waterloo, Canada. He researches various aspects of information security. His work, with students, has won paper-awards at the ACM Symposium on Access Control Models and Technologies 2013 and 2015, and the Use-nix Security Symposium 2013.

▷ **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.**

Short Papers

Connecting the Dots: Privacy Leakage via Write-Access Patterns to the Main Memory

Tara Merin John¹, Syed Kamran Haider¹,
Hamza Omar¹, and Marten van Dijk

Abstract—Data-dependent access patterns of an application to an untrusted storage system are notorious for leaking sensitive information about the user's data. Previous research has shown how an adversary capable of monitoring both read *and* write requests issued to the memory can correlate them with the application to learn its sensitive data. However, information leakage through *only* the write access patterns is less obvious and not well studied in the current literature. In this work, we demonstrate an actual attack on power-side-channel resistant Montgomery's ladder based modular exponentiation algorithm commonly used in public key cryptography. We infer the complete 512-bit secret exponent in ~ 3.5 minutes by virtue of just the write access patterns of the algorithm to the main memory. In order to learn the victim algorithm's write access patterns under realistic settings, we exploit a compromised DMA device to take frequent snapshots of the application's address space, and then run a simple differential analysis on these snapshots to find the write access sequence. The attack has been shown on an Intel Core(TM) i7-4790 3.60GHz processor based system. We further discuss a possible attack on McEliece public-key cryptosystem that also exploits the write-access patterns to learn the secret key.

Index Terms—Secure processors, privacy leakage, write-access patterns, montgomery ladder exponentiation

1 INTRODUCTION

USERS' data privacy is becoming a major concern in computation outsourcing in the current cloud computing world. Numerous secure processor architectures (e.g., XOM [1], [2], TPM+TXT [3], Aegis [4], Intel-SGX [5] etc.) have been proposed for preserving data confidentiality and integrity during a remote secure computation. The user sends his encrypted data to a secure processor where it is decrypted and computed upon in a tamper-proof environment, and finally the encrypted results of the computation are sent back to the user.

While the secure processors provide sufficient levels of security against *direct* attacks, most of these architectures are still vulnerable to *side-channel* attacks. For instance, XOM and Aegis architectures are vulnerable to control flow leakage via address bus snooping [6], [7], [8]. Similarly, Intel-SGX, being a strong candidate in secure architectures, is vulnerable to side-channel attacks via a compromised OS[9].

Zhuang et al. [10] showed that although the data in the main memory of the system can be encrypted, the access patterns to the memory could still leak privacy. An adversary who is able to monitor both read *and* write accesses made by an application can relate this pattern to infer secret information of the application. For example, Islam et al. [11] demonstrated that by observing accesses to an encrypted e-mail repository, an adversary can infer as much as 80 percent of the search queries. This, however, is a very strong adversarial model which, in most cases, requires direct physical access to the memory address bus. In cloud computing, for example, this requires the cloud

service itself to be untrusted. The challenging requirements posed by the above mentioned strong adversarial model leads one to think that applications are vulnerable to privacy leakage via memory access patterns only if such a strong adversary exists, i.e., one capable of monitoring both read *and* write accesses.

In this paper, we counter this notion by demonstrating privacy leakage under a significantly weaker adversarial model. In particular, we show that an adversary capable of monitoring *only* the write access patterns of an application can still learn a significant amount of its sensitive information. Hence, in the model of computation outsourcing to a secure processor discussed earlier, even if the cloud service itself is trusted, a remote adversary is still able to steal private information if the underlying hardware does not protect against leakage from write access patterns.

We present a real attack on the famous Montgomery's ladder technique [12] commonly used in public key cryptography for modular exponentiation. Exponentiation algorithms, in general, are vulnerable to various timing and power side-channel attacks [13], [14], [15]. Montgomery's ladder performs redundant computations as a countermeasure against power side-channel attacks (e.g., simple power analysis [16]). However, by monitoring the order of write accesses made by this algorithm, one can still infer the secret *exponent* bits.

In our weaker adversarial model, since we cannot directly monitor the memory address bus, we learn the pattern of write accesses by taking frequent memory snapshots. For this purpose, we exploit a compromised Direct Memory Access device (DMA¹) attached to the victim computer system to read the application's address space in the system memory [17], [18], [19]. Clearly, any two memory snapshots only differ in the locations where the data has been modified in the latter snapshot. In other words, comparing the memory snapshots not only reveals the fact that write accesses (if any) have been made to the memory, but it also reveals the exact locations of the accesses which leads to a precise access pattern of memory writes.

Our experimental setup uses a PCI Express to USB 3.0 adapter attached to the victim system, alongside an open source application called PCILeech [20], as the compromised DMA device. We implement the Montgomery's ladder for exponentiation of a 128 byte message with a 64 byte (512 bits) secret exponent [21]. Through our attack methodology, we are able to infer all 512 secret bits of the exponent in just 3 minutes and 34 seconds on average.

Although our experimental setup utilizes a wired connection to a USB 3.0 port on the victim system for DMA, Stewin et al. demonstrated that DMA attacks can also be launched *remotely* by injecting malware to the dedicated hardware devices, such as graphic processors and network interface cards, attached to the host platform [22]. Therefore, our attack methodology allows even remote adversaries to exploit the coarse grained side-channel information obtained by memory snapshots to infer the secret data. Hence, this effort opens up new research avenues to explore efficient countermeasures to prevent privacy leakage under remote secure computation.

2 BACKGROUND

2.1 Exponentiation Algorithms

Exponentiation algorithms have central importance in cryptography, and are considered to be the back-bone of nearly all the public-key cryptosystems. Although numerous exponentiation algorithms have been devised, algorithms for constrained devices

1. DMA grants full access of the main memory to certain peripheral buses, e.g., FireWire, Thunderbolt etc.

• The authors are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06279.
E-mail: {tara.john, syed.haider, hamza.omar, marten.van_dijk}@uconn.edu.

Manuscript received 18 May 2017; revised 14 Nov. 2017; accepted 14 Nov. 2017. Date of publication 4 Dec. 2017; date of current version 18 Mar. 2020.

(Corresponding author: Syed Kamran Haider.)

Digital Object Identifier no. 10.1109/TDSC.2017.2779780

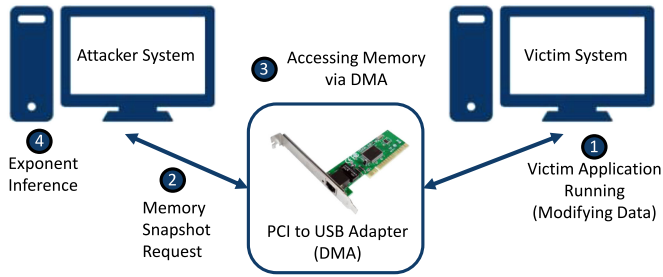


Fig. 1. Our adversarial model: The attacker system takes snapshots of the victim’s DRAM via the PCI adapter to infer the secret key.

are scarcely restricted to the square-and-multiply algorithms. RSA algorithm, used in, e.g., Diffie-Hellman key agreement, is a commonly used exponentiation algorithm which performs computation of the form $y = g^k \bmod n$, where the attacker’s goal is to find the secret key k . The commonly used square-and-multiply implementation of this algorithm is shown in Algorithm 1. For a given input g and a secret key k , Algorithm 1 performs multiplication and squaring operations on two local variables R_0 and R_1 for each bit of k starting from the most significant bit down to the least significant bit.

Algorithm 1. RSA - Left-to-Right Binary Algorithm

Inputs: $g, k = (k_{t-1}, \dots, k_0)_2$ **Output:** $y = g^k$

Start:

```

1:  $R_0 \leftarrow 1; R_1 \leftarrow g$ 
2: for  $j = t - 1$  downto  $0$  do
3:    $R_0 \leftarrow (R_0)^2$ 
4:   if  $k_j = 1$  then  $R_0 \leftarrow R_0 R_1$  end if
5: end for
return  $R_0$ 

```

Notice that the conditional statement on line 1 of Algorithm 1 executes based on the value of secret bit k_j . Such conditional branches result in two different power and timing spectra of the system for $k_j = 0$ and $k_j = 1$, hence leaking the secret key k over the timing/power side-channels. Similar attacks [21] have leaked 508 out of 512 bits of an RSA key by using branch prediction analysis (BPA). Thus, the attack-prone nature of RSA algorithm (Algorithm 1) poses a need for an alternate secure algorithm.

2.2 Montgomery’s Power Ladder Algorithm

Montgomery Power Ladder [12] shown in Algorithm 2 performs exponentiation without leaking any information over power side-channel. Regardless of the value of bit k_j , it performs the same number of operations in the same order, hence producing the same power footprint for $k_j = 0$ and $k_j = 1$. Notice, however, that the specific order in which R_0 and R_1 are updated in time depends upon the value of k_j . E.g., for $k_j = 0$, R_1 is written first and then R_0 is updated; whereas for $k_j = 1$ the updates are done in the reverse order. This sequence of write access to R_0 and R_1 reveals to the adversary the exact bit values of k . In this paper, we exploit this vulnerability in a real implementation of Montgomery ladder to learn the secret key k .

3 THE PROPOSED ATTACK

3.1 Adversarial Model

Consider a computer system that is continuously computing exponentiations of the form $y = g_i^k$ for the given inputs g_i using the same secret exponent k according to Algorithm 2. We call this system the *victim* system. All the data stored in the main memory of this system is encrypted. Let there be a compromised DMA device

(e.g., a PCI-to-USB adapter) connected to the victim system through which an attacker system can read the whole main memory of the victim as shown in Fig. 1. The attacker system, however, is limited in its ability to successively read the victim’s memory by the data transfer rate of the underlying DMA interface. The adversary’s goal is to find the key k by learning the application’s write pattern through frequent snapshots of the victim system’s memory. The victim system used in our attack comes with a *write-through* cache configuration enabled by default. As a result, any write operations performed by the application are immediately propagated through the memory hierarchy down to the untrusted DRAM. Furthermore, we assume that the victim application receives all the inputs g_i in a batch and continuously produces the corresponding cipher texts such that the physical memory region allocated to the application during successive encryptions remains the same. In other words, the application is not relocated to a different physical address space by the OS throughout the attack. Such use cases can be found in the applications that require computing signatures of large files.

Algorithm 2. Montgomery Power Ladder Algorithm

Inputs: $g, k = (k_{t-1}, \dots, k_0)_2$ **Output:** $y = g^k$

Start:

```

1:  $R_0 \leftarrow 1; R_1 \leftarrow g$ 
2: for  $j = t - 1$  downto  $0$  do
3:   if  $k_j = 0$  then  $R_1 \leftarrow R_0 R_1; R_0 \leftarrow (R_0)^2$ 
4:   else  $R_0 \leftarrow R_0 R_1; R_1 \leftarrow (R_1)^2$ 
5:   end if
6: end for
return  $R_0$ 

```

3.2 Attack Outline

Given the above mentioned setting, we proceed with our attack methodology as follows: First, a full scan of the victim’s memory is performed to identify the physical address space allocated to the victim’s application. Since the adversary requires victim application’s memory snapshots at a high frequency, it is infeasible for him to always read the full victim memory because of the data transfer rate being the frequency limiting factor. Once the address space is identified, the next step is to identify the two memory regions allocated to each of the local variables R_0 and R_1 (cf. Algorithm 2) within the victim application’s address space. This allows any observed change in either of these two regions to be linked with an update to the variables R_0 and R_1 respectively. Finally, the updates in R_0 and R_1 memory regions are observed via frequent snapshots for a period of one complete encryption, and the order of these updates is linked back to Algorithm 2 to learn the key k . We explain these steps in detail in the following sections.

3.3 Step 1: Application’s Address Space Identification

Since the application is supposed to be continuously updating its data (e.g., variables R_0, R_1), its address space can be identified by finding the memory regions which are continuously being updated. Algorithm 3 shows this process at an abstract level. The whole of the victim system’s memory space is divided into M blocks, each of some reasonable size B (say a few megabytes). Two subsequent snapshots of each block $m \in M$ are compared with each other through COMPAREMATCH procedure. It is a heuristic based process which searches for a specific pattern of updates between the two snapshots which potentially represents the application’s footprint. For example, a sequence of two modified consecutive 64 byte cache lines followed by a few unmodified cache lines and then further two modified consecutive cache lines would potentially represent the two 128 byte regions for R_0 (first two cache lines) and R_1 (last two cache lines). Finally, a set S of all

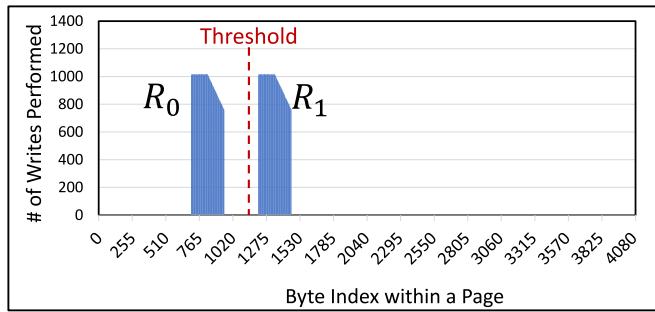


Fig. 2. A histogram of # of writes to individual bytes in the victim's memory page. A clear distinction is shown between the regions corresponding to variables R_0 and R_1 .

those memory blocks which show the specific update sequence searched by COMPAREMATCH is returned. This algorithm is iteratively repeated until a reasonably small set of memory block(s) (e.g., one 4 kB page) is identified which is expected to contain the victim application's address space.

Algorithm 3. Victim App's Address Space Identification

Inputs: M : Set of memory blocks to scan.

Output: S : Set of application's memory block(s).

Start:

- 1: $S = \emptyset$ ▷ Initially empty set.
- 2: **for** $m \in M$ **do** ▷ Scan each block.
- 3: $s_1 = \text{TAKE_SNAPSHOT}(m)$
- 4: $s_2 = \text{TAKE_SNAPSHOT}(m)$
- 5: **if** COMPAREMATCH(s_1, s_2) **then**
- 6: $S = S \cup m$
- 7: **end if**
- 8: **end for**
- 9: **return** S

3.4 Step 2: Distinguishing Local Variables R_0 and R_1

Once the application's memory space is found, we need to link two distinct regions within this address space to the variables R_0 and R_1 in order to determine the key bits from the order of their updates. For this purpose, a set V of n snapshots of the application's space is computed as shown in Algorithm 4. Notice that n is large enough to cover one full encryption period. The COMPUTETHRESHOLD procedure computes a histogram of the updates performed inside the application's memory over all the snapshots of set V . Fig. 2 shows one such histogram for a 4kB page of victim's memory. It can be seen that almost all the updates are performed at two distinct regions spanning over only a few cache lines within the page. These two regions correspond to the variables R_0 and R_1 respectively.² The *inactive* region between R_0 and R_1 represents a threshold which is later used by CORRELATE procedure to determine whether a change in two successive memory snapshots corresponds to an update in R_0 or R_1 etc.

3.5 Step 3: Inferring the Secret Key

After computing the set of snapshots V and the threshold Th , we enter the final phase of inferring the secret key (starting from step 4 in Algorithm 4). Up to this point, the sequence V contains pairs of snapshots that represent changes in R_0 and R_1 , and also the pairs which represent no change, as shown in Fig. 3. The reason why some pairs do not show any change is because our snapshot frequency is higher than the rate at which the application updates its

2. We can tell whether R_0 or R_1 comes first in the memory layout from the declaration order of these variables in the actual implementation of the exponentiation algorithm (cf. line 2 in Algorithm 2).

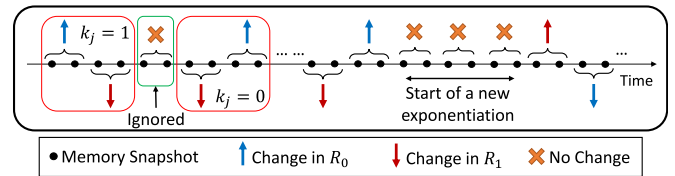


Fig. 3. Inferring the secret key via observing the sequence of snapshots and the changes in variables R_0 and R_1 . The pairs of snapshots which do not show any change are ignored.

data. This allows us to learn the write access pattern at a fine granularity.

Algorithm 4. Pseudo Code for the Second Phase of Attack

Input: S : Application's memory space. (from Algorithm 3);

n : # of snapshots to cover one full encryption period.

Output: k : Application's secret key.

Start:

- 1: $V = (s_1, \dots, s_n) \mid s_i = \text{TAKE_SNAPSHOT}(S), 1 \leq i \leq n$
- 2: $Th = \text{COMPUTE_THRESHOLD}(V)$
- 3: $W = \emptyset, k = (0, \dots, 0)$
- 4: $V = \text{REMOVE_UNCHANGED_SNAPSHOTS}(V)$
- 5: **for** $i = 1$ **to** $|V| - 1$ **do**
- 6: $R_{x_i} = \text{CORRELATE}(s_i, s_{i+1}, Th) \quad \triangleright x_i \in \{0, 1\}$
- 7: $W = W \cup R_{x_i}$
- 8: **end for**
- 9: $i = 1, j = 0$
- 10: **for** $(R_{x_i}, R_{x_{i+1}}) \in W$ **do**
- 11: **if** $R_{x_i} = R_0$ and $R_{x_{i+1}} = R_1$ **then** $k_j = 1$
- 12: **else if** $R_{x_i} = R_1$ and $R_{x_{i+1}} = R_0$ **then** $k_j = 0$
- 13: **end if**
- 14: $i = i + 2; j = j + 1$
- 15: **end for**
- return** k

In order to learn the write access pattern, first the pairs of unchanged snapshots from the sequence V are removed by the procedure REMOVEUNCHANGEDSNAPSHOTS V . The resulting sequence V only contains pairs which always represent a change, either in R_0 or R_1 . Now, each pair of two successive snapshots is correlated to an update in either R_0 or R_1 by CORRELATE procedure using the threshold computed earlier.

As mentioned earlier, Montgomery ladder algorithm performs computations upon local variables, where the order of variable updates is based on the secret exponent bits (cf. Algorithm 2). Therefore, judging from the order of updates made in R_0 and R_1 , each pair of updates $(R_{x_i}, R_{x_{i+1}}) \in W$ is linked back to the corresponding value of the secret key bit k_j as shown in Fig. 3. As the set W contains the history of all the updates to R_0 and R_1 for a complete encryption, therefore all the key bits can be inferred through the above mentioned process.

3.6 Modern Systems with Write-Through Caches

It would be surprising for the reader to know that even in this age and time, there exist certain commercial computer systems which, to the best of our knowledge, are shipped with configurations such that their *write-back* caching capability is disabled by default, and enabling this feature requires a nominal fee to be paid to the CPU vendor. In our experimental setup, we use one such computer system (based on Intel Core(TM) i7-4790) as our victim machine whose further specifications are presented in Section 4.1. The caching hardware configuration of this system read by running `lshw` Linux command is shown in Fig. 4. It can be seen that the L3 cache, although *write-back*, is currently disabled which only leaves a hierarchy of two *write-through* caches L1 and L2.

```

*-cache:0
description: L1 cache
physical id: 27
slot: L1-Cache
size: 256KiB
capacity: 256KiB
capabilities: internal write-through
unified
*-cache:1
description: L2 cache
physical id: 29
slot: L2-Cache
size: 1MiB
capacity: 1MiB
capabilities: internal write-through
unified
*-cache:2 DISABLED
description: L3 cache
physical id: 2a
slot: L3-Cache
size: 8MiB
capacity: 8MiB
capabilities: internal write-back
unified

```

Fig. 4. Output of `lshw` command showing caches' configuration.

We like to remind the reader that this is the default configuration how this system was shipped to us by the vendor, and the L3 has not been disabled by the authors. Considering the fact that thousands of such other systems must have been shipped by the vendor to other customers, it is alarming to think about that those customers are vulnerable to privacy leakage via the proposed attack while not having even a slight idea about it.

4 ATTACK DEMONSTRATION

4.1 Experimental Setup

Our experiment setup uses two computer systems, one being the attacker and the other being the victim. The victim system is *DELL XPS 8700*, comprising of *Intel Core(TM) i7-4790 3.60 GHz* processor that uses *Ubuntu 14.04.3 LTS* operating system with a *Linux kernel 3.19.0-43-generic*, and has 16 GB of main memory. The attacker machine is a *64-bit Windows 10* based system having 8 GB of main memory. A PCI adapter module, called *USB Evaluation Board* [23], is connected to the victim via the PCI-Express slot and acts as a compromised DMA device (cf. Fig. 1). This DMA device, together with PCILeech software [20], allows the attacker to monitor victim's memory and/or take its snapshots. To implement our attack, the PCILeech software has been extended to first find the application's address space in the victim's memory (cf. Algorithm 3), and then attack the identified address space to infer the secret key (cf. Algorithm 4). The above mentioned attacking algorithms run *while* the victim application is executing.

The victim system has a BIOS version *A11* which supports *write-through* enabled L1 and L2 caches while disabling the L3 cache by default. Besides caches, any data modifications in the *register file* should also be propagated to the DRAM. The register file (usually only a few hundred bytes) is used by the processor to temporarily hold the operands and results during computations. We have written our own C++ implementation of the Montgomery ladder based exponentiation algorithm³ for large input sizes (128 Bytes or more)

3. Available at GitHub (<https://github.com/meriniamo/RSA-Montgomery-Ladder-Implementation>)

that runs on the victim system. Since our implementation uses multiple temporary variables and several function calls for proper execution of the algorithm, the large "active" working set of the application cannot fit into the register file and results in *register spills*. Hence, any updates made by the application are immediately propagated—through register file and caches—down to the main memory as each multiplication/squaring write operation is performed. Notice that, the attack can be mitigated if the "active" working set can fit into the register file, however for realistic applications (such as the one under discussion), it is mostly not possible. Section 4.2 explains the step by step details about how the attack is launched.

4.2 Experimental Results

In order to take memory snapshots via PCI module and the PCILeech software, the attacker first needs to load a kernel module into the victim system via the PCI module itself. Notice that the attacker does not require any extra privileges to do so. We use the following command via PCILeech software to load the kernel into victim's DRAM. When the kernel is loaded, an address is spitted out by the software, which shows where the module resides in the victim's memory. Loading the kernel into memory is a rapid process and takes only a few milliseconds to complete the process.

```

D:\>pcileech kmdload -kmd LINUX_X64
KMD: Successfully loaded at 0x1b54a000

```

In the meantime, the Montgomery's ladder exponentiation algorithm is run on the victim machine using a 128 byte (1024 bits) message along with a secret key of 64 bytes (512 bits). With the application running and the kernel module loaded into victim's memory, we proceed to find the potential regions in the DRAM which are being accessed frequently by taking multiple snapshots. To retrieve these snapshots, we issue the *pagefind* command shown below which uses the loaded kernel module's address to access the victim's full memory.

We integrated the *pagefind* command into the PCILeech software to iteratively find regions getting modified persistently. *pagefind* narrows down the selected regions to a single page by constantly monitoring and comparing the changes being made, and returns the address of the page where application's array data structures are defined. This step corresponds to *Application's Address Space Identification* phase of the attack (cf. Algorithm 3) and is the most time consuming phase. To read the whole memory, comparing their respective snapshots and narrowing down to a single page of 4KB from 16GB search space takes ~3 minutes and 30 seconds.

```

D:\>pcileech pagefind -kmd 0x1b542000
Page Finding: Successful.
Total_Time : 210199 Milliseconds
Victim Page Address : 0xd271c000

```

As shown above, from the first phase we retrieve the address of the page where application's data structures are stored. Proceeding towards our second and third step namely *Distinguishing Local Variables* and *Inferring the Secret Key* (cf. Sections 3.4, 3.5), we use another integrated command *pageattack*. It first takes a predefined number of snapshots of the application page provided by the first step, and distinguishes the message (R_1) and algorithm result (R_0) from the rest of the stale data, residing on the memory page. It then uses the order of changes in R_0 and R_1 to infer the secret key.

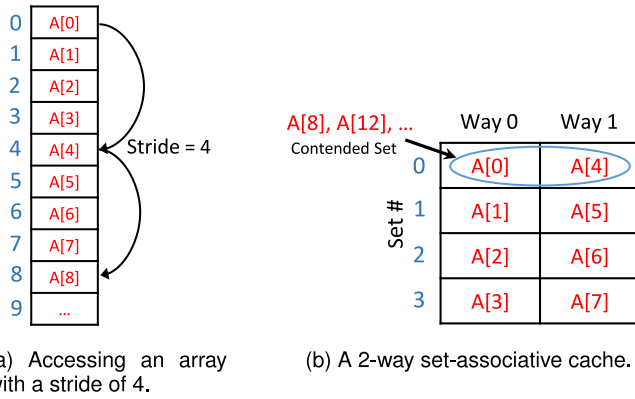


Fig. 5. A strided memory access pattern causing contention on a single cache set.

```
D:\>pcileech pageattack -min 0xd271c000
Attack Successful.
Total_Time = 3596 Milliseconds
Inferred Key is:
1a 4b 28 41 e6 27 d4 7d
72 c3 40 79 be 1f 6c 35
ca 3b 58 b1 96 17 04 ed
22 b3 70 e9 6e 0f 9c a5
7a 2b 88 21 46 07 34 5d
d2 a3 a0 59 1e ff cc 15
2a 1b b8 91 f6 f7 64 cd
82 93 d0 c9 ce ef fc 85
```

This final step takes ~ 3.6 seconds to complete and returns the complete 512 bit secret key learned from only the write access patterns. Combining the times associated with all the attack phases, the total attack time comes out to be ~ 3 minutes, 34 seconds.

5 LEAKAGE UNDER CACHING EFFECTS

In view of our proposed attack on Montgomery ladder based exponentiation, the updates to the application data should always be available in the DRAM of the victim system before an attacker issues a memory snapshot request. This is only possible if the victim system has *write-through* enabled cache hierarchy or the caches are disabled altogether. Whereas, on the other hand, modern processors typically consist of large on-chip *write-back* caches where the updates to application's data are only be visible in DRAM once the data is evicted from the last level cache (LLC). Thus in the attack proposed in Section 3, the caching effects are not catered for, which introduce 'noise' to the precise write-access sequence inferred earlier, hence making the attacker's job difficult. A possible workaround to deal with such caching effects is to collect several 'noisy' sequences of memory snapshots and then run correlation analysis on them to learn the precise write-access pattern. Furthermore, if the adversary is also a user of the same computer system, it can flush the system caches frequently to reduce the noise in write-access sequence even further.

Another (more efficient) attack scenario under write-back caches would be when the application has a strided memory access pattern that causes contention over the cache sets, and hence forces its own data to be evicted to make room for the new data in the cache. In the following section, we discuss how such a strided memory access pattern can lead to evictions to the DRAM which could potentially leak private information.

5.1 Memory Striding and Cache Set Contention

A strided memory access pattern is the one where each request to the memory is for the same number of bytes, and the access pointer

is incremented by the same amount between each request. An array accessed with a stride of exactly the same size as the size of each of its elements results in accessing contiguous locations in the memory. Such access patterns are said to have a stride value of 1. Fig. 5a shows a *non-unit* striding access pattern in which the elements 0, 4, 8, 12, \dots of an array A are accessed. This access pattern has a stride value of 4.

Consider a simple system which has a 2-way set-associative *write-back* cache with a total capacity of 8 cache lines, as shown in Fig. 5b. The strided access pattern from Fig. 5a accesses every 4 i th element of the array A , where $i = 0, 1, 2, \dots$. Assuming that each element of A is of size equal to the cache line size, for a simple *modulus based* cache hash function, the elements $A[0], A[4], A[8], \dots$ are mapped to the same set causing contention over *set 0*. Since the cache associativity is only 2, this access sequence causes evictions from the cache when both ways of set 0 contain valid cache lines. Similarly, elements $A[1], A[5], A[9], \dots$ map to *set 1*, and this access sequence will cause evictions from set 1, and so on. In other words, such write-access sequences are still propagated almost immediately to the next level in the memory hierarchy (e.g., DRAM) even under write-back caches, which could potentially leak information. This is an artifact of the cache implementation combined with the striding access pattern of the application.

It must be noted that, not all evictions result in updates to the main memory. Typically, only *dirty* cache lines, caused by data writes, evicted from the cache are propagated to the main memory. *Clean* evictions from the cache are simply discarded resulting in no change in the main memory since it already contains a clean copy of the data.

Assume that an application generates two distinguishable striding write-access patterns that result in contention at two different cache sets, leading to evictions from the cache. Consequently, the resulting write-access access sequence will be revealed to an adversary who is capable of monitoring changes in the main memory, potentially resulting in privacy leakage.

5.2 Striding Application: Gaussian Elimination

In Section 5.1 we discussed, using a toy example, how a strided access pattern can lead to information leakage. Now we present a realistic example which has such a striding access pattern, and later in Section 5.3 we show how such a pattern can be exploited to learn private information. We consider the application of *Gaussian Elimination* of large binary matrices carrying substantial amount of information. Clearly, these large matrices cannot fit into the caches, therefore there will be cache evictions as a result of Gaussian elimination operations.

Gaussian elimination a.k.a. row reduction is a method for solving system of linear equations by the use of matrices in the form $Ax = B$. Row reduction is done by doing a series of elementary row operations which modify the matrix until it forms an upper triangular matrix, i.e., elements underneath the main diagonal are zeros. Different types of elementary row operations include swapping two rows, multiplying a row by a non-zero number and adding a multiple of one row to another. The upper triangular matrix formed out of these operations will be in row echelon form. When the leading coefficient (pivot) in each row is 1, and every column containing the leading coefficient has zeros elsewhere, the matrix is said to be in reduced row echelon form. The Gaussian elimination algorithm consists of two processes, one being *forward elimination* that converts the matrix to row echelon form and the other is *backward substitution* that calculates values of the unknowns. These processes result in solving the linear equation.

Gauss-Jordan elimination uses a similar approach for finding the inverse of a matrix. For a $n \times n$ square matrix S , elementary row operations can be applied to reduce the matrix into reduced echelon form, and furthermore, for computing the matrix inverse if

$$\begin{array}{ccc}
 \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} & \xrightarrow{\left[\begin{array}{l} \leftarrow + \\ \leftarrow + \\ \leftarrow + \end{array} \right]} & \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \text{[Step 1]} & & \text{[Step 2]} \\
 \\
 \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \xrightarrow{\left[\begin{array}{l} \leftarrow + \\ \leftarrow + \\ \leftarrow + \end{array} \right]} & \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \text{[Step 3]} & & \text{[Step 4]}
 \end{array}$$

 Fig. 6. The gaussian elimination process on a 4×4 binary matrix.

it exits. Initially, the $n \times n$ identity matrix I is augmented to the right of S , forming a $n \times 2n$ block matrix $[S|I]$. Now, upon applying the row operations, the left block can be reduced to the identity matrix I if S is invertible. This gives S^{-1} which is the right block of the final matrix. In a nutshell, we continue performing row operations until $[S|I]$ becomes $[I|S^{-1}]$.

Consider that the matrix under elimination is stored in a *column-contiguous* manner in the computer system's main memory. In other words, each column occupies a contiguous chunk of memory equal to the column size, after which the next column resides, and so on. When consecutive elements of a row of this matrix are accessed during a row operation, the corresponding memory access pattern results in a striding sequence, where the stride length is equal to the column size. If the stride length is such that it creates contention on particular cache sets corresponding to particular rows, this would reveal the modified row, which in turn could potentially leak the binary matrix itself (cf. Section 5.3).

5.3 Attacking McEliece Public-Key Cryptosystem

McEliece public key cryptosystem [24], an asymmetric encryption algorithm, uses an error correcting code for a description of the private key. This encryption uses a fast and efficient decoding algorithm, namely a Goppa code and hides the structure of the code by transformation of the generator matrix. This transformation yields the public key and the structure of the Goppa code together with the transformation parameters, which further provides the trapdoor information. For a linear code C , generator matrix G , random invertible matrix S and random permutation matrix P , the matrix $G^* = SGP$ is made public while P , G , and S form the private key. A message m is encrypted along with a random error vector using the equation $c = mG^* + e$, where c refers to the ciphertext. In the decryption process, we compute $c^* = cP^{-1}$, decode c^* to m^* by the decoding algorithm, and lastly compute $m = m^*S^{-1}$. Notice, that S is a private binary matrix whose inverse is used to recover the message m . Any system carrying out this encryption/decryption process could either store the matrix inverse (for better performance) or calculate the inverse during the run time. However, the latter could lead to the leakage of the binary matrix via write-access patterns during the inverse computation. In this section we will demonstrate how performing Gauss-Jordan elimination [25] on the binary secret matrix S could lead to its complete exposure as a consequence of cache striding and cache set contention as shown in Section 5.1.

For the ease of demonstration we consider a 4×4 binary matrix. The elements stored in the main memory are column contiguous. We assume that each element of the matrix is *cache line aligned* for performance reasons. In other words, each element is stored in a unique cache line in order to avoid false sharing within a cache line. Considering a system with a 2-way 4-set associative cache, each row of the matrix is mapped to one cache set due to the cache structure (cf. Fig. 5b). The elimination process to obtain the inverse of the binary matrix $S_{4 \times 4}$ is shown step by step in Fig. 6. After these row operations, we obtain an identity matrix $I_{4 \times 4}$.

$$C_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{\left[\begin{array}{l} \leftarrow + \\ \leftarrow + \end{array} \right]} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = S_2$$

$$C_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{\left[\begin{array}{l} \leftarrow + \\ \leftarrow + \\ \leftarrow + \end{array} \right]} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = S_3$$

$$C_4 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{\left[\begin{array}{l} \leftarrow + \\ \leftarrow + \\ \leftarrow + \end{array} \right]} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{\left[\begin{array}{l} \leftarrow + \\ \leftarrow + \\ \leftarrow + \end{array} \right]} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = S_4$$

 Fig. 7. Back substitution process to recover secret binary matrix S .

In each of the above 4 steps, the corresponding pivot row is added to another row or rows. For example, in **Step[1]** row 1 is added to row 2 and row 4. Similarly, row 2 is added to row 3 in **Step[2]**. As a row operation is performed, the elements of the target row are modified and result in cache line evictions, since accessing a whole row causes contention over the corresponding set it is mapped to, and a cache set can only store 2 elements of a row at a time. For instance, in **Step[1]**, row 2 and row 4 are modified causing contention and evictions from set 1 and 3 respectively. Consequently, an adversary can learn the identifier of the row being updated during each row operation by monitoring the address space in which any updates take place, and then linking it back to the row number. Now, by definition of the elimination algorithm, all column elements corresponding to rows that undergo addition operations can be inferred as 1s, and the remaining ones as 0s. Hence, in each of **Step[1]**, **Step[2]**, **Step[3]** and **Step[4]**, we infer the corresponding pivot column to be $C_1 = \{1, 1, 0, 1\}$, $C_2 = \{0, 1, 1, 0\}$, $C_3 = \{1, 1, 1, 0\}$ and $C_4 = \{1, 0, 1, 1\}$ respectively.

Notice that C_2 , C_3 , and C_4 obtained in the above steps show the respective intermediate forms of the corresponding columns of S during the elimination process. These values, however, can be used to recover the original column values of matrix S through *back substitution* process, as shown in Fig. 7. In this process, each column C_i undergoes the row operations performed (and inferred) in each of the steps **Step[i - 1]**, **Step[i - 2]**, ..., **Step[1]**, precisely in this order. For example, C_2 undergoes addition of row 1 to rows 2 and 4, while C_3 performs addition of row 2 to row 3 along with the addition of row 1 to rows 2 and 4. Upon completion of the back substitution process, the complete secret matrix S is recovered by the adversary.

6 DISCUSSION

6.1 Countermeasures for Our Attack

One approach to prevent privacy leakage via write-access pattern leveraging DMA based attacks, as demonstrated in this paper, could be to block certain DMA accesses through modifications in the DRAM controller. However, this approach poses complexity in terms of how to determine which accesses to allow and which ones to block. Furthermore, it requires this 'extended' DRAM controller to be included in the trusted computing base (TCB) of the system which is undesirable.

Another strong candidate is Oblivious RAM [26], [27] which is a well known technique to prevent privacy leakage via memory access patterns. Although, current so called *fully functional* ORAMs, e.g., Path ORAM [26], which obfuscate both read and write patterns, offer a possible countermeasure (at a cost of performance penalty) against the attack we demonstrated. However, the extra

protection (read pattern obfuscation) offered by these approaches is an overkill for current attack scenario and incurs redundant performance penalties.

A better alternative could be a *write-only* ORAM [28], [29], [30] which only obfuscates write-access patterns and not the reads. This technique offers far better performance than a fully functional ORAM under such weaker adversarial models. It has been shown that *write-only* ORAM has an optimal asymptotic communication overhead of $O(1)$ as compared to the fully functional ORAM schemes, which are asymptotically $\Omega(\log n)$ [28].

7 CONCLUSION

Privacy leakage via purely write-access patterns is less obvious and not extensively studied in the current literature. We demonstrate a real attack on Montgomery's ladder based modular exponentiation algorithm and infer the secret exponent by just learning the write access patterns of the algorithm to the main memory. We adapt the traditional DMA based exploits to learn the application's write access pattern in a reasonable time. Our attack takes just 3 minutes and 34 seconds to learn 512 secret bits from a typical Linux based victim system. A possible attack on McEliece public-key cryptosystem has also been presented. We discuss some possible countermeasures to prevent such attacks. Further research towards developing efficient countermeasures is left as future work.

ACKNOWLEDGMENTS

This work was partially supported by NSF grant CNS-1413996 for MACS: A Modular Approach to Cloud Security. T. M. John and S. K. Haider contributed equally to this work. This work is an extension of an abstract submission, accepted at a poster in HOST 2017, which received "the best poster award".

REFERENCES

- [1] D. Lie, J. Mitchell, C. Thekkath, and M. Horowitz, "Specifying and verifying hardware for tamperresistant software," in *Proc. IEEE Symp. Security Privacy*, 2003, Art. no. 166.
- [2] D. Lie, C. Thekkath, and M. Horowitz, "Implementing an untrusted operating system on trusted hardware," in *Proc. 19th ACM Symp. Operating Syst. Principles*, 2003, pp. 178–192.
- [3] D. Grawrock, *The Intel Safer Computing Initiative: Building Blocks for Trusted Computing*. Santa Clara, CA, USA: Intel Press, 2006.
- [4] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas, "AEGIS: Architecture for Tamper-Evident and Tamper-Resistant Processing," in *Proc. Int. Conf. Supercomput.*, Jun. 2003, pp. 160–171.
- [5] F. McKeen, et al., "Innovative instructions and software model for isolated execution," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2013, Art. no. 10.
- [6] G. E. Suh, D. Clarke, B. Gassend, M. V. Dijk, and S. Devadas, "Efficient memory integrity verification and encryption for secure processors," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2003, pp. 339–350.
- [7] J. Yang, Y. Zhang, and L. Gao, "Fast secure processor for inhibiting software piracy and tampering," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2003, pp. 351–360.
- [8] B. Gassend, G. E. Suh, D. Clarke, M. V. Dijk, and S. Devadas, "Caches and hash trees for efficient memory integrity verification," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2003, pp. 295–306.
- [9] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 640–656.
- [10] X. Zhuang, T. Zhang, and S. Pande, "Hide: An infrastructure for efficiently protecting information leakage on the address bus," *ACM SIGPLAN Notices*, vol. 39, no. 11, pp. 72–84, 2004.
- [11] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. 2nd Netw. Distrib. Syst. Security Symp.*, 2012, vol. 20, Art. no. 12.
- [12] M. Joye and S. M. Yen, "The montgomery powering ladder," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.* 2002, pp. 291–302.
- [13] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Annual International Cryptology Conference*. Berlin, Germany: Springer, 1996.
- [14] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Comput. Netw.*, vol. 48, no. 5, pp. 701–716, 2005.
- [15] L. Goubin, "A refined power-analysis attack on elliptic curve cryptosystems," in *Public Key Cryptography Workshop*. Berlin, Germany: Springer, 2003, pp. 199–211.
- [16] S.-M. Yen, L.-C. Ko, S. Moon, and J. Ha, "Relative doubling attack against montgomery ladder," in *International Conference on Information Security and Cryptology*. Berlin, Germany: Springer, 2005, pp. 117–128.
- [17] D. Aumaitre and C. Devine, "Subverting windows 7 x64 kernel with DMA attacks," *Sogeti ESEC Lab*, (2010). [Online]. Available: <http://esec-lab.sogeti.com/static/publications/10-hitbamsterdam-dmaattacks.pdf>
- [18] D. Maynor, "DMA: Skeleton key of computing && selected soap box rants," *CanSecWest*, 2005. [Online]. Available: <http://cansecwest.com/core05/DMA.ppt>
- [19] B. Böck and S. B. Austria, "Firewire-based physical security attacks on windows 7, EFS and bitlocker," *Secure Bus. Austria Lab'09*. [Online]. Available: https://dl.packetstormsecurity.net/papers/win/windows7_firewire_physical_attacks.pdf
- [20] U. Frisk, "Pcileech: Direct memory access attack software." (2016). [Online]. Available: <https://github.com/ufrisk/pcileech>
- [21] O. Aciçmez, C. K. Koç, and J.-P. Seifert, "On the power of simple branch prediction analysis," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, 2007, pp. 312–320.
- [22] P. Stewin and I. Bystrov, "Understanding dma malware," in *Conference on Detection of Intrusions and Malware & Vulnerability Assessment*. Berlin, Germany: Springer, 2012.
- [23] I. BPlus Technology, "Usb3380 evaluation board." (2011). [Online]. Available: <http://www.bplus.com.tw/Adapter/USB3380EVB.html>
- [24] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *Coding Thv*, vol. 4244, pp. 114–116, 1978.
- [25] A. Bogdanov, M. C. Mertens, C. Paar, J. Pelzl, and A. Rupp, "Smith-a parallel hardware architecture for fast gaussian elimination over $GF(2)$," in *Proc. Workshop Special-Purpose Hardw. Attacking Cryptographic Syst.*, (2006). [Online]. Available: http://ei.ruhr-uni-bochum.de/media/crypto/veroeffentlichungen/2011/01/29/sharcs2006_matrix.pdf
- [26] E. Stefanov, et al., "Path ORAM: An extremely simple oblivious ram protocol," in *Proc. ACM Comput. and Commun. Security Conf.*, 2013, pp. 299–310.
- [27] L. Ren, X. Yu, C. W. Fletcher, M. Van Dijk, and S. Devadas, "Design space exploration and optimization of path oblivious ram in secure processors," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 571–582, 2013.
- [28] E.-O. Blass, T. Mayberry, G. Noubir, and K. Onarlioglu, "Toward robust hidden volumes using write-only oblivious ram," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 203–214.
- [29] L. Li and A. Datta, "Write-only oblivious ram-based privacy-preserved access of outsourced data," *Int. J. Inform. Security*, vol. 16, pp. 1–20, 2013.
- [30] S. K. Haider and M. van Dijk, "Flat ORAM: A simplified write-only oblivious ram construction for secure processor architectures," arXiv preprint arXiv:1611.01571, 2016. [Online]. Available: <https://arxiv.org/pdf/1611.01571.pdf>

Known Sample Attacks on Relation Preserving Data Transformations

Emre Kaplan¹, Mehmet Emre Gursoy²,
Mehmet Ercan Nergiz¹, and Yuçel Saygin¹

Abstract—Many data mining applications such as clustering and k -NN search rely on distances and relations in the data. Thus, distance preserving transformations, which perturb the data but retain records' distances, have emerged as a prominent privacy protection method. In this paper, we present a novel attack on a generalized form of distance preserving transformations, called relation preserving transformations. Our attack exploits not the exact distances between data, but the relationships between the distances. We show that an attacker with few known samples (4 to 10) and direct access to relations can retrieve unknown data records with more than 95 percent precision. In addition, experiments demonstrate that simple methods of noise addition or perturbation are not sufficient to prevent our attack, as they decrease precision by only 10 percent.

Index Terms—Data mining, security and privacy, data transformations, known sample attacks, protection

1 INTRODUCTION

DATA privacy is a fundamental challenge, and various methods of data perturbation and obfuscation have been proposed to overcome this challenge. Among these, distance preserving transformations (DPTs) have gained significant attention due to their simplicity and ability to retain good accuracy for many data mining applications such as clustering and classification.

In this paper, we propose an attack on exact and approximate DPTs. Our attack relies on distance *relations* rather than actual distances themselves. That is, instead of asking “How close exactly are X and Y ?”, we ask “Is Y closer to X than Z ?”. The prior can be easily harmful since it may reveal Y 's exact location, but the latter seems naive. A real-world application is through a location based social network: User X performs a proximity search by requesting a list of nearby users. Y is among the returned list of nearby users, but Z is not. Then, X infers that his distance to Y is smaller than his distance to Z . A DPT of this data enables the same inference to be made, since even though the data is transformed, distances between records are preserved. Furthermore, even if the distances are not exactly preserved, there can be an arbitrarily high probability of preserving distance relations, e.g., distance between X - Y is smaller than X - Z . We call such transformations that preserve distance relations (but not necessarily exact distances) relation preserving transformations (RPTs).

We show, through a novel attack, that the seemingly naive query involving distance relations can be harmful when used in a known sample attack. That is, if an attacker has a few known samples (that he is certain will be part of the private data) and can obtain distance relations concerning them and other unknown records in the data, then the attacker will be able to infer the contents of the unknown records with high accuracy and precision.

- E. Kaplan, M. E. Nergiz, and Y. Saygin are with the Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul 34956, Turkey. E-mail: {emrekaplan, ercann, ysaygin}@sabanciuniv.edu.
- M. E. Gursoy is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332. E-mail: memregursoy@gatech.edu.

Manuscript received 3 Feb. 2017; revised 22 July 2017; accepted 21 Sept. 2017. Date of publication 5 Oct. 2017; date of current version 18 Mar. 2020.

(Corresponding author: Yuçel Saygin.)

Digital Object Identifier no. 10.1109/TDSC.2017.2759732

To better position our work within the literature, we first survey previous work on DPTs and attacks on DPTs. Then, we list our contributions and the primary characteristics of our attack.

Work on DPTs. Some data mining algorithms rely on distances between records rather than the records themselves. These algorithms work equally well without knowing the private data, but instead observing its distance matrix or transformed version (using a DPT). In [1], Chen and Liu show that the following are examples of such algorithms: k -NN classifiers, kernel methods, Support Vector Machines using polynomial, radial basis and neural network kernels, linear classifiers, and some clustering and regression models.

The three traditional techniques for DPTs are *translations*, *reflections* and *rotations* [2], [3]. Translations shift records a constant distance in parallel directions. Reflections map records to their mirror images in fixed-dimensional space. Rotations rotate all records by a fixed angle called the rotation angle. In more recent work, Huang et al. [4] present FISIP, in which they propose DPTs that preserve first order and second order sums and inner products of records. In [5], Giannella et al. argue that by tuning the parameters of random projection transformations, one can ensure arbitrarily high probabilities of distance preservation. They point to [6] for preliminary results in this direction.

Attacks on DPTs. Attacks on DPTs were first considered by Liu et al. [7], and later studied in [8], [9], [10] and [5]. We report the main differences between our work and these previous works. For a more general survey, we refer the reader to [11].

In [7], Liu et al. develop two attacks on DPTs, one where the attacker has a set of known samples, and one where the attacker has a set of known input-output pairs. In the latter, the attacker knows the true values of perturbed records, and this assumption is significantly stronger than what we assume in our work. We assume a known sample attack, which coincides with Liu et al.'s first attack. However, while their attack requires a significant number of known samples (e.g., 5 percent of the data, which can correspond to hundreds of samples) to be successful, we can achieve same rates of disclosure with only 4-6 samples. Thus, our attack is significantly more realistic and practical. In [9], Turgay et al. extend the attacks in [7] by assuming that the attacker has a distance matrix of the private data. They assume that the actual distribution of the data is known by the attacker, and develop attacks based on principal component analysis. In comparison, our attack works without knowledge of the data distribution.

In [8], Guo and Wu assume that the attacker has a set of known samples and aims to retrieve the remaining data using Independent Component Analysis (ICA). Their attack applies to arbitrary transformations, but their experiments imply that the attack requires approximately 500-1,000 known samples to reach the precision of our attack. In [10], Chen et al. confirm that although DPTs (in particular, rotation and geometric perturbation) are useful for outsourced data mining, they are not resilient against ICA-based attacks. Therefore Chen et al. propose algorithms that compute ICA-resilient perturbations with additive noise, which meet a desired level of privacy guarantee and attack resilience.

Most recently, Giannella et al.'s attack in [5] achieves precision most similar to ours. However, their approach is probabilistic whereas ours is deterministic. That is, they identify records' original locations with certain confidence, whereas (without the presence of noise) we can identify locations always with 100 percent confidence. In addition, even though Giannella et al.'s attack as well as the attacks discussed above require DPTs, our attack runs on RPTs, which is a more relaxed and generalized set of transformations.

Contributions. In this paper, we propose an attack on RPTs. The highlights and distinctive features of our attack are as follows:

TABLE 1
Creating the Distance Matrix of a Spatial Database

ID	Coordinates	r_1	r_2	r_3	r_4
r_1	(34.0, 122.6)	0	68.1	77.4	95.6
r_2	(13.1, 57.8)	68.1	0	12.1	160
r_3	(2.5, 51.9)	77.4	12.1	0	170.8
r_4	(98.4, 193.2)	95.6	160	170.8	0

(a) Database \mathcal{D} with four records

(b) Distance matrix of \mathcal{D}

TABLE 2
A relation Preserving Transformation of \mathcal{D}

ID	Coordinates	r'_1	r'_2	r'_3	r'_4
r'_1	(32.7, 123.6)	0	65.9	73.5	99.5
r'_2	(14.5, 60.3)	65.9	0	8.4	161.2
r'_3	(8.8, 54.1)	73.5	8.4	0	169.4
r'_4	(100.0, 196.9)	99.5	161.2	169.4	0

(a) Transformed database \mathcal{D}'

(b) Distance matrix of \mathcal{D}'

(1) We generalize from DPTs to RPTs: RPTs only preserve the distance relationships while transforming the data, whereas DPTs preserve exact distances. We show that DPTs are a subset of RPTs, and therefore our attack directly applies to DPTs. (2) We base our attack solely on a set of known samples and relations. As such, the attacker need not even observe the transformed data or its distance matrix. Also, the attacker need no apriori knowledge of data distribution. (3) The attack is applicable to the distance matrix publication model [9] as well as the perturbed data publication model [8], since both models trivially leak distance relations. (4) We achieve high precision with very small and realistic known sample sizes, e.g., 2-6 known samples. Concretely, when attacking a target record with only 4 known samples, the attacker can reduce the data space by 96 percent, leaving only the remaining 4 percent as to where the target record could be located. Thus, our approach is more effective than many of the previous works. (5) We propose extensions, such as space discretization and voting, for ease of implementation and noise resilience respectively. Experiments show that such extensions are feasible and effective.

2 PRELIMINARIES AND NOTATION

The data owner's private database \mathcal{D} is denoted with transcript $\mathcal{D}(r_1, \dots, r_n)$, where $r_i \in \mathcal{D}$ are the records in \mathcal{D} . We make no assumptions regarding the structure or type of data contained in \mathcal{D} , apart from the ability to map \mathcal{D} to a m -dimensional Euclidean space \mathbb{R}^m . As such, we view each r_i as a *point* in Euclidean space, and use the terms *record* and *point* interchangeably.

Our starting point is pairwise distances between points in Euclidean space: As their name implies, distance preserving transformations (DPTs) preserve pairwise distances. Pairwise distances between elements $r_i, r_j \in \mathcal{D}$ can be computed using commonly used distance metrics, e.g., Minkowski (p -norm) distance, Euclidean distance, etc. [12] Without loss of generality, we assume that Euclidean distance is used.

Definition 1 (Euclidean distance). Let x and y be two data points in \mathbb{R}^m , with coordinates $x = (x^1, \dots, x^m)$ and $y = (y^1, \dots, y^m)$. We say that the Euclidean distance between x and y is: $\delta(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^m (x^i - y^i)^2}$, where $\|\cdot\|$ denotes the L^2 -norm.

For help in giving concrete examples, we introduce the notion of distance matrix (also known as the dissimilarity matrix [2]) that captures pairwise distances between points.

Definition 2 (Distance matrix). The distance matrix of a database $\mathcal{D}(r_1, \dots, r_n)$ is an $n \times n$, symmetric, real-valued matrix M such that $M_{i,j} = M_{j,i} = \delta(r_i, r_j)$.

For example, let \mathcal{D} be a geospatial database containing (X, Y) coordinates of 2D data points. A sample database \mathcal{D} is shown in Table 1 a. \mathcal{D} 's distance matrix is given in 1 b. As an example, we compute one of the entries in the distance matrix: $M_{1,2} = \delta(r_1, r_2) = \sqrt{(34.0 - 13.1)^2 + (122.6 - 57.8)^2} = 68.1$.

Definition 3 (Distance Preserving Transformation). A function $\mathcal{T} : \mathbb{R}^m \rightarrow \mathbb{R}^d$ is a distance preserving transformation (DPT) if for all $x, y \in \mathbb{R}^m$, $\delta(x, y) = \delta(\mathcal{T}(x), \mathcal{T}(y))$.

Let \mathcal{D} be the data owner's private database. Instead of releasing \mathcal{D} , for privacy protection the data owner first perturbs \mathcal{D} using a DPT \mathcal{T} and then releases the perturbed data $\mathcal{D}' = (\mathcal{T}(r_1), \dots, \mathcal{T}(r_n))$. By definition, \mathcal{T} preserves pairwise distances between records, and thus the distance matrix M is constant before and after \mathcal{T} . We note that $\mathcal{T} : \mathbb{R}^m \rightarrow \mathbb{R}^d$ where we do not require $m = d$. That is, we allow \mathcal{T} to change the dimensionality of the data. This increases the scope of transformations by covering the likes of principle component analysis, singular value decomposition and kernel functions.

In this paper we consider a broader type of transformations that we call *relation preserving transformations*. Such transformations allow pairwise distances (hence, the distance matrix) to change, but only in a way that the relative order of the distances is conserved. That is, assuming M is the distance matrix of the original data and M' is the distance matrix of the transformed data, if $M_{i,j}$ is greater than [less than] $M_{k,l}$, $M'_{i,j}$ must be greater than [less than] $M'_{k,l}$. Relation preserving transformations have the desirable property that similar data mining results can be obtained although exact pairwise distances between records are not revealed. For example, a record's k nearest neighbors do not change, therefore k -NN classification on transformed data would produce the same output as if it were run on the original data.

Definition 4 (Relation Preserving Transformation). A function $\mathcal{S} : \mathbb{R}^m \rightarrow \mathbb{R}^d$ is a relation preserving transformation (RPT) if for $x, y \in \mathbb{R}^m$, $z, t \in \mathbb{R}^d$ and for arithmetic comparison operators $op \in \{<, >, =\}$, $\delta(\mathcal{S}(x), \mathcal{S}(y)) op \delta(\mathcal{S}(z), \mathcal{S}(t))$ if and only if $\delta(x, y) op \delta(z, t)$.

Theorem 1. Every DPT is relation preserving.

Proof. Let $\mathcal{T} : \mathbb{R}^m \rightarrow \mathbb{R}^d$ be a DPT. Since \mathcal{T} is distance preserving, for all $x, y, z, t \in \mathbb{R}^m$, $\delta(\mathcal{T}(x), \mathcal{T}(y)) = \delta(x, y)$ and $\delta(\mathcal{T}(z), \mathcal{T}(t)) = \delta(z, t)$. Therefore, trivially for any comparison operator op , $\delta(\mathcal{T}(x), \mathcal{T}(y)) op \delta(\mathcal{T}(z), \mathcal{T}(t))$ if and only if $\delta(x, y) op \delta(z, t)$. \square

The goal of this paper is to attack RPTs. We use Theorem 1 to show that DPTs are a subset of RPTs, therefore attacks on RPTs are applicable to DPTs (whereas the literature mostly focuses on attacks on DPTs). Notice that the converse of Theorem 1 is not true: A RPT is not necessarily distance preserving. For example, let \mathcal{D}' in Table 2 a be obtained via transforming \mathcal{D} in Table 1 a in a relation preserving manner. One can verify that the pairwise order of distances in Table 2 b is the same as in Table 1, but none of the distances actually stay the same.

Definition 5 (Relation Retrieval Function). For a database \mathcal{D} and arbitrary $r_A, r_B, r_C, r_D \in \mathcal{D}$, the relation retrieval function \mathcal{R} is defined as:

$$\mathcal{R}_{\mathcal{D}}((r_A, r_B), (r_C, r_D)) = \begin{cases} -1 & \text{if } \delta(r_A, r_B) < \delta(r_C, r_D) \\ 0 & \text{if } \delta(r_A, r_B) = \delta(r_C, r_D) \\ 1 & \text{if } \delta(r_A, r_B) > \delta(r_C, r_D) \end{cases}$$

\mathcal{R} is essentially a function that captures the relations between records' distances in a dataset. From Definitions 3 and 4, we have $\mathcal{R}_{\mathcal{D}} = \mathcal{R}_{\mathcal{D}'}$, where \mathcal{D}' is obtained from \mathcal{D} using a DPT or RPT.

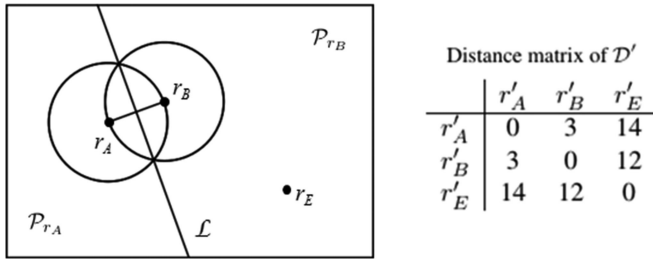


Fig. 1. Sample 2-dimensional database \mathcal{D} with three records. Actual locations of records in \mathbb{R}^2 (on the left) and the distance matrix published after transformation (on the right).

3 ATTACK ALGORITHM

We propose a novel strategy to attack RPTs. We assume that the following information is available to the attacker:

- *Outputs of the relation retrieval function, $\mathcal{R}_{\mathcal{D}'}$.* That is, the attacker knows the *relations* between distances in the transformed dataset. Clearly, the attacker can obtain this if he is given either \mathcal{D}' or \mathcal{M}' . Assuming that the attacker has $\mathcal{R}_{\mathcal{D}'}$ is a more relaxed assumption than assuming he has \mathcal{D}' or \mathcal{M}' , but the latter is the prevalent assumption in related work (e.g., [2], [9]).
- *A set of known samples.* The attacker has a sample of records $r_i \in \mathcal{D}$. That is, the attacker knows where each r_i maps to in the original \mathbb{R}^m space (prior to transformation).

Known-sample attacks are popular in the literature [5], [7], [9], [12]. There are multiple ways in which an attacker can obtain a set of known samples, e.g., the attacker may know that his and a few other friends' information is in the data, or may be able to inject a record into the data. Notice that our attack makes no assumptions regarding the underlying transformation function used or the transformed output. That is, we do not require the attacker to obtain input-output pairs or any output points from the transformation function \mathcal{S} .

In Section 3.1, we present our attack on 2D data so that we can easily illustrate it with examples. In Section 3.2, we formalize the attack and generalize it to arbitrary dimensions.

3.1 Intuitive Explanation of the Attack

We introduce our attack using the example in Fig. 1. Suppose that database \mathcal{D} is 2-dimensional (i.e., records are in \mathbb{R}^2), and let r_A, r_B in \mathcal{D} be the known samples of the attacker. Say that the goal of the attacker is to find the position of r_E , i.e., locate r_E in \mathbb{R}^2 space. Let \mathcal{M}' be the distance matrix that is published after a RPT \mathcal{S} is applied to \mathcal{D} .

Observation 1. If $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = -1$, then in the original dataset r_E must be located outside the circle with center r_A and radius $\delta(r_A, r_B)$.

Observation 2. If $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = 0$, then in the original dataset r_E must be located on the circle with center r_A and radius $\delta(r_A, r_B)$.

Observation 3. If $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = 1$, then in the original dataset r_E must be located within the area enclosed by the circle with center r_A and radius $\delta(r_A, r_B)$.

Given the known samples r_A, r_B , and \mathcal{R} , the attacker iteratively prunes the search space while searching for r_E . Observations 1, 2 and 3 demonstrate one way of pruning. The main idea is to compare the distance between the two known samples (r_A and r_B), and the distance between the target (r_E) and the known samples after the transformation is applied. The relation preservingness of the transformation allows the attacker to make inferences on the original dataset and prune out those portions in \mathbb{R}^2 that r_E cannot be

located in. In all of the observations above, we compare $\delta(r_A, r_B)$ with $\delta(r_A, r_E)$, however $\delta(r_A, r_B)$ can be also compared with $\delta(r_B, r_E)$ which would result in circles centered at r_B with radius $\delta(r_A, r_B)$. The procedure can also be repeated for every pair of samples the attacker has (we considered only one pair (r_A, r_B) so far).

These observations are exemplified in Fig. 1. Since in the distance matrix we have $\delta(r'_A, r'_B) = 3$ is less than $\delta(r'_A, r'_E) = 14$, we have $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = -1$. The attacker that knows this information performs the following pruning: He draws the circle centered at r_A and radius equal to $\delta(r_A, r_B)$. Let $\delta(r_A, r_B) = 2$. Then, r_E cannot be within this circle, since r_E is further away from r_A than r_B . Next, since we have $\delta(r'_B, r'_E) = 12$, following the same reasoning, we draw the circle centered at r_B and radius equal to $\delta(r_A, r_B) = 2$. r_E must also be outside this circle, since r_E is further away from r_B than r_A .

We now present a second type of pruning. For the two known data samples r_A and r_B , let \mathcal{L} denote the perpendicular bisector of the hypothetical line connecting r_A and r_B . (Given the locations of r_A and r_B , it is trivial to draw both the hypothetical line and its perpendicular bisector.) As seen in Fig. 1, \mathcal{L} divides the search space into two portions. Let \mathcal{P}_{r_A} denote the portion that contains r_A and \mathcal{P}_{r_B} denote the portion that contains r_B .

Observation 4. If $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = 1$, then r_E must be located in \mathcal{P}_{r_B} .

Proof (Sketch). From the definition of \mathcal{R} , we have $\delta(\mathcal{S}(r_A), \mathcal{S}(r_E)) > \delta(\mathcal{S}(r_B), \mathcal{S}(r_E))$. Since \mathcal{S} is relation preserving, this implies $\delta(r_A, r_E) > \delta(r_B, r_E)$. \mathcal{L} is a line containing points that are equidistant to r_A and r_B . All points $X \in \mathcal{P}_{r_B}$ have the property $\delta(r_B, X) < \delta(r_A, X)$, whereas points Y on \mathcal{L} satisfy $\delta(r_B, Y) = \delta(r_A, Y)$ and points $Z \in \mathcal{P}_{r_A}$ satisfy $\delta(r_A, Z) < \delta(r_B, Z)$. Hence, r_E is in \mathcal{P}_{r_B} . \square

Observation 5. If $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = 0$ then r_E must be located on \mathcal{L} .

Observation 6. If $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = -1$ then r_E must be located in \mathcal{P}_{r_A} .

The second type of observations we make (Observations 4, 5 and 6) examine the distance between the target r_E and the two known samples (i.e., $\delta(r_A, r_E)$ and $\delta(r_B, r_E)$). Again, this process can be repeated for every pair of known samples.

We exemplify using Fig. 1. We have $\delta(r'_A, r'_E) = 14$ and $\delta(r'_B, r'_E) = 12$. Thus, $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = 1$. This enables the attacker to infer that r_E is closer to r_B than it is to r_A . Then, he draws the perpendicular bisector \mathcal{L} and locates $r_E \in \mathcal{P}_{r_B}$.

At this point we would like to underline two characteristics of our attack: (1) The pruning process is fully deterministic, i.e., the attacker is 100 percent confident that pruned areas may not contain the target data point. (2) We are placing constraints on where r_E can/cannot be located in the original dataset, not the transformed dataset. The attacker's goal is to locate r_E in the original data space, not in the transformed space.

3.2 Attack Formalization

In this section we formalize our attack and generalize it to n -dimensional space \mathbb{R}^n , where $n \geq 2$.

Definition 6 (Hypersphere). In n -dimensional Euclidean space, a hypersphere $S_{C,r}$ is defined by a fixed center point $C \in \mathbb{R}^n$ and a radius r , and denotes the set of points in the n -dimensional space that are at distance r from C . That is, each point $X(X_1, X_2, \dots, X_n)$ on $S_{C,r}$ satisfies: $r^2 = \sum_{i=1}^n (X_i - C_i)^2$.

Definition 7 (Hyperball). In n -dimensional Euclidean space, given a hypersphere $S_{C,r}$, the hyperball $B_{C,r}$ denotes the space enclosed by $S_{C,r}$. $B_{C,r}$ is said to be closed if it includes $S_{C,r}$ and open otherwise.

For example, circles are hyperspheres in 2-dimensional space. A circle with center $C(3, 4)$ and radius 7 would be characterized by the equation: $(x - 3)^2 + (y - 4)^2 = 49$. Then, the open hyperball $B_{(3,4),7}$ specifies the area enclosed by this circle, excluding those points that are on the circle. This is given by the equation: $(x - 3)^2 + (y - 4)^2 < 49$.

Definition 8 (Equidistant Hyperplane). In n -dimensional Euclidean space, the locus of points equidistant from two points $A, B \in \mathbb{R}^n$ is a hyperplane H_{AB} that contains all points P that satisfy the equality $\|P - A\| = \|P - B\|$.

For example, let $A, B \in \mathbb{R}^3$ have the following coordinates: $A(-2, 1, 4), B(0, 6, 3)$. We say that P has the coordinates $P(x, y, z)$ and solve $\|P - A\| = \|P - B\|$. According to euclidean distance, this builds the equation: $\sqrt{(x+2)^2 + (y-1)^2 + (z-4)^2} = \sqrt{(x-0)^2 + (y-6)^2 + (z-3)^2}$. After simple arithmetic, we obtain: $4x + 10y - 2z = 24$, which is the equation of the equidistant hyperplane H_{AB} .

Definition 9 (Half-Space). A half-space is either of the two parts into which a hyperplane divides the n -dimensional euclidean space. A half-space is said to be closed if it includes the hyperplane, and open otherwise.

Half-spaces are often specified using linear inequalities derived from the hyperplane that separates them. For the previous example, the two open half-spaces of \mathbb{R}^3 are given by the equations:

$$\begin{aligned} \mathcal{P}_A : 4x + 10y - 2z < 24 \\ \mathcal{P}_B : 4x + 10y - 2z > 24 \end{aligned}$$

It is clear to see that since H_{AB} is the hyperplane that is equidistant to A and B , one of the half-spaces will contain point A whereas the other will contain B . We call these half-spaces \mathcal{P}_A and \mathcal{P}_B respectively.

We present our attack strategy in Algorithm 1. We have the following inputs: The universe U is a collection of all possible points that may exist in the database. The universe often has boundaries that are dictated by the semantics of the underlying database, e.g., the boundaries of a dimension *age* could be 0 and 110. Or, if the database contains the locations or spatial information regarding people living in a particular city, the universe is bounded by the borders of that city. $\mathcal{R}_{\mathcal{D}}$ captures the distance relations after a RPT. The attacker has a set of legitimate known samples (i.e., all samples are within the universe U). We describe the attack assuming the attacker would like to compromise the location of one target record r_E , but the attack can be run on an arbitrary record. We specify the identifier of the target record, E , as one of our inputs.

Initially, we say that r_E can be anywhere in the universe, by setting the search space variable (denoted by s) to U . For every pair of known samples, we iteratively prune the search space several times using the observations made in the previous section. Here, pruning refers to deleting certain geometric objects, or areas that do not intersect with a given geometric object, from the search space. To make the algorithm easier to follow, we give the specific observations that lead to each of the pruning operations. On lines 4-5 we apply Observation 1, on lines 6-7 we apply Observation 3, and on lines 8-9 we apply Observation 2. These three steps are based on the distances between (r_A, r_B) and (r_A, r_E) . We then apply the same approach to the distances between (r_A, r_B) and (r_B, r_E) to obtain the three steps between lines 10-16: On lines 11-12 we apply Observation 1, on lines 13-14 we apply Observation 3, and on lines 15-16 we apply Observation 2. Afterwards, between lines 17-23, we apply Observations 4, 5 and 6. Specifically, on lines 18-19 we apply Observation 4, on lines 20-21 we apply Observation 6, and on lines 22-23 we apply Observation 5. The final output of

the attack is the region of the universe that has not been pruned, i.e., the area where r_E must be located in.

Algorithm 1. Attack for Locating a Target Record using Distance Relations and Known Samples

Input: U : data space and its boundaries,
 $\mathcal{R}_{\mathcal{D}}$: distance relations of the transformed data,
 $K = \{r_1, \dots, r_n | r_i \in U\}$: set of known samples,
 E : an identifier to denote the target record r_E

Output: $U' \subseteq U$: portion of the universe where the target record is located

- 1: $s \leftarrow U$
- 2: **for** each pair $(r_A, r_B) \in K$ **do**
- 3: Build the hypersphere $S_{r_A, \delta(r_A, r_B)}$ and open hyperball $B_{r_A, \delta(r_A, r_B)}$
- 4: **if** $\mathcal{R}_{\mathcal{D}}((r_A, r_B), (r_A, r_E)) = -1$ **then**
- 5: $s \leftarrow s - (B_{r_A, \delta(r_A, r_B)} \cup S_{r_A, \delta(r_A, r_B)})$
- 6: **else if** $\mathcal{R}_{\mathcal{D}}((r_A, r_B), (r_A, r_E)) = 1$ **then**
- 7: $s \leftarrow s \cap B_{r_A, \delta(r_A, r_B)}$
- 8: **else**
- 9: $s \leftarrow s \cap S_{r_A, \delta(r_A, r_B)}$
- 10: Build the hypersphere $S_{r_B, \delta(r_A, r_B)}$ and open hyperball $B_{r_B, \delta(r_A, r_B)}$
- 11: **if** $\mathcal{R}_{\mathcal{D}}((r_A, r_B), (r_B, r_E)) = -1$ **then**
- 12: $s \leftarrow s - (B_{r_B, \delta(r_A, r_B)} \cup S_{r_B, \delta(r_A, r_B)})$
- 13: **else if** $\mathcal{R}_{\mathcal{D}}((r_A, r_B), (r_B, r_E)) = 1$ **then**
- 14: $s \leftarrow s \cap B_{r_B, \delta(r_A, r_B)}$
- 15: **else**
- 16: $s \leftarrow s \cap S_{r_B, \delta(r_A, r_B)}$
- 17: Build the equidistant hyperplane $H_{r_A r_B}$ and resulting open half-spaces $\mathcal{P}_{r_A}, \mathcal{P}_{r_B}$
- 18: **if** $\mathcal{R}_{\mathcal{D}}((r_A, r_E), (r_B, r_E)) = 1$ **then**
- 19: $s \leftarrow s \cap \mathcal{P}_{r_B}$
- 20: **else if** $\mathcal{R}_{\mathcal{D}}((r_A, r_E), (r_B, r_E)) = -1$ **then**
- 21: $s \leftarrow s \cap \mathcal{P}_{r_A}$
- 22: **else**
- 23: $s \leftarrow s \cap H_{r_A r_B}$
- 24: **return** s

3.3 Implementation Considerations

Discretization. As can be seen in Algorithm 1, our attack involves many union, intersection and difference operations. These are non-trivial to implement in continuous n -dimensional space. For the sake of reproducibility, we comment on the specifics of our implementation.

We implement the attack by discretizing the search space: We assume that the universe U is made up of uniform n -dimensional cells. Smaller the cells are, higher the total number of cells and finer the granularity of the attack will be. However, due to the increased number of cells, execution time will also be higher. We take a defensive approach when we prune cells, that is, in each pruning decision we prune only those cells that can be *completely* pruned off the search space. For example, consider Fig. 2. When we prune the half-space \mathcal{P}_{r_A} , we only prune those cells that are completely contained by \mathcal{P}_{r_A} . For all borderline cells (e.g., those that lie on \mathcal{L}) we keep them instead of pruning them. As such, we guarantee that we never over-prune, e.g., if in Fig. 2 we prune cell Y then we would have over-pruned by removing the portion that is to the right of \mathcal{L} , which includes area that r_E could actually be located in. This would violate the correctness of our attack. On the other hand, by not pruning cell Y we also keep the portion in cell Y that is to the left of \mathcal{L} , which we are certain that r_E is not located in. We ideally would like to prune the latter portions off, but since our cells were too coarse (i.e., too big) in this case, we could not do so. On the

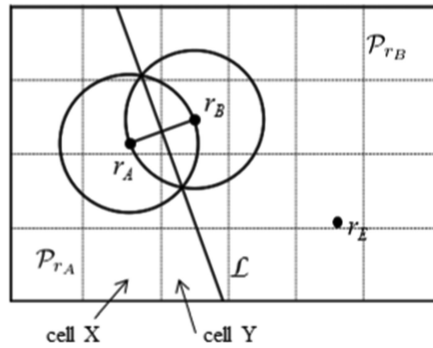


Fig. 2. Discretization of the universe using uniform 2-dimensional cells.

other hand, we safely prune *cell X* since the whole cell lies within \mathcal{P}_{r_A} .

Complexity Analysis. We analyze the time complexity of our attack given the implementation details above. Let the universe U be n -dimensional, and for the sake of simplicity let us assume that each dimension has the same length (i.e., difference between the endpoints of its boundaries) of $\Omega(U)$. Further, let us divide the search space into uniform cells with side lengths of c . Then, we have $(\frac{\Omega(U)}{c})^n$ cells in total.

Algorithm 1 executes its main loop (between lines 2-23) a total of $\binom{|K|}{2}$ times. Within the loop, it builds two hyperspheres and one hyperplane. Given the equations of these geometrical structures, constructing them is trivial and can be done in constant time. However, for each structure, it performs exactly one space pruning operation, which depends on computing the intersection of each cell and the constructed structures. We incorporate a factor \mathcal{I} for computing intersections, which depends heavily on the programming language and geometry libraries used. Then, the overall time complexity of our attack is $O(\binom{|K|}{2} \cdot (\frac{\Omega(U)}{c})^n \cdot \mathcal{I})$.

As implied above, the attack is exponential in n (number of dimensions) and $|K|$ (number of known samples). Although this may seem prohibitive, we note that: (i) The attack is typically not executed in real-time, and therefore there are no strict efficiency requirements. For example, if the sensitive information being attacked is a victim's home location, since this does not change frequently, it is not a burden to wait for a few hours to obtain the attack outcome. (ii) Generally both n and $|K|$ are low, e.g., in our experimental setting we use 2D location data from location-based social networks with few known samples, so that $n = 2$ and $|K| \in [2, 10]$. With $|K| = 4$, our implementation (in Java) can execute the attack in 8 seconds, using a laptop with a single 2.2 GHz processor and 8 GB memory.

Noise Resilience. One of the prominent techniques in data privacy is *additive perturbation*, which adds noise to the published information. Thus, it is interesting to make our attack resilient to the addition of noise. Note that noise addition will likely destroy the relation preservingness of a transformation, and the correctness of our attack can no longer be guaranteed. That is, given data space U the attack may output that r_E resides in space U' , but in fact r_E resides in $U - U'$.

Algorithm 1 is not resilient to noise, since it prunes a region of the search space immediately when one pair decides that the region should be pruned. For instance, let (r_C, r_D) be a pair that decides to prune *cell X* when searching for target record r_E . Algorithm 1 would immediately prune *cell X* from the search space variable s and proceed. This is not problematic in the noise-free case. However, in the noisy case, $\mathcal{R}_{\mathcal{D}}$ may be imperfect due to the added noise. Thus, (r_C, r_D) 's decision to prune *cell X* could be wrong.

To account for these cases, we implemented a *voting mechanism*. For each cell, we keep track of the pairs of records that have *voted*

in favor of pruning that cell. If, at any point, the number of votes for pruning *cell X* exceeds an input voting threshold, we prune *X* off the search space. That is, let $t \in [0, 1]$ denote the voting threshold and $|K|$ be the number of known samples. If $t \times |K|$ samples vote in favor of pruning *X*, then *X* will be pruned. We use the voting mechanism only when the data is noisy (i.e., the transformation is not guaranteed to be relation preserving) or the answers to $\mathcal{R}_{\mathcal{D}}$ are noisy (e.g., untruthful).

Further Improvements. We discuss potential improvements that increase the practicality of our attack under two settings: *limited time* and *limited access*.

In the limited time setting, we assume that the attack must be executed in a short amount of time, e.g., the attacker wants to know where the victim *currently* is, in a dynamic city environment. The exponential complexity of the attack might be a problem, therefore we propose two improvements to overcome this problem: pre-computation with known samples, and multi-granularity grids. For pre-computation, we assume that the attacker's known samples stay constant over time, or follow a probability distribution known by the attacker. For example, if the known samples are the attacker's friends in a social network, then the attacker would know their probable locations (e.g., work, home, frequent visits). Given these for each pairs of known samples, the attacker can pre-compute $S_{r_A, \delta(r_A, r_B)}$, $S_{r_B, \delta(r_A, r_B)}$ and H_{r_A, r_B} , since these do not rely on the victim's record r_E . Then, the attacker determines which cells would be pruned depending on the answers of \mathcal{R} , and hardcodes them into the attack to improve efficiency. The second improvement, multi-granularity grids, proposes to replace the single-level uniform grid with hierarchically organized multiple-layer grids (e.g., $8 \times 8 \rightarrow 4 \times 4 \rightarrow 2 \times 2$). Then, if a cell in the topmost grid with largest cells (i.e., 2×2) can be pruned, we immediately infer all cells in the finer-granularity grids (i.e., 8×8) that are covered by the larger cell can also be pruned. The use of more sophisticated and data-dependent indexing structures, e.g., quad-trees and kd-trees, are left to future work.

In the limited access setting, we assume that the data is outsourced to an external service, and the attacker can issue a bounded number of relation retrieval queries \mathcal{R} due to their monetary cost or privacy countermeasures implemented by the external service. Then, we need to study methods to maximize the effectiveness of our attack under this constraint. For this purpose, we propose strategically selecting the pairs of known samples (on line 2 of Alg. 1) that lead to highest amount of expected pruning. Specifically, we suggest that the attacker chooses a balanced combination of: (i) Nearby known samples, i.e., (r_A, r_B) such that $\delta(r_A, r_B)$ is as small as possible. In this case, the circles drawn in Fig. 1 are small. Then, if Observation 3 holds, the attacker reduces the search space into a small circle, which increases his precision. (ii) Distant known samples, i.e., (r_C, r_D) such that $\delta(r_C, r_D)$ is as large as possible. In this case, the circles drawn in Fig. 1 are large. Then, if Observation 1 holds, the attacker can prune large circles off the search space. A combination of these two extremes should give the attacker an increased amount of expected pruning.

4 EXPERIMENTS

4.1 Experiment Setup

Consider the following practical application of our attack: The data owner runs a mobile service and collects private location check-ins of users. This data is shared with third parties after a RPT. Since the attacker and a few close friends of his are users of the mobile service, the attacker knows their check-in locations and this constitutes his set of known samples. Then, the goal of the attacker is to infer the locations of remaining users (whose locations he cannot directly observe, since, e.g., they are not in his social circle). Motivated by this scenario, we use two 2D real-life location datasets in our experiments, which are as follows:

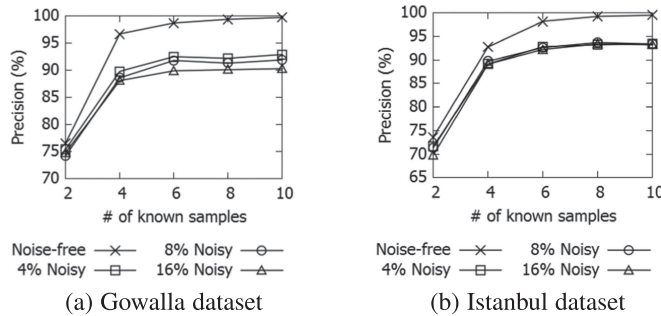


Fig. 3. Precision in noise-free and noisy scenarios.

Gowalla Dataset [13]. Gowalla was a location-based social networking website where users shared their locations by checking-in. A total of 6,442,890 check-ins over the period of February 2009 and October 2010 were collected and made available.¹ This dataset was recently used in other privacy related works as well. From the Gowalla data, we extracted those check-ins made in New York.

Istanbul Dataset. We collected location data from 60,000 vehicles in Istanbul, Turkey using a vehicle tracking system. From this data, we extracted the last known locations of 200 randomly chosen vehicles and formed our experimental dataset. The data consists of the vehicle ids, latitude and longitude coordinates.

We use our grid-based implementation explained in Section 3.3. We pick the grid cell size such that the area of each cell is 0.01km^2 . We believe that locating an individual within such a small area is reasonable and challenging considering the entire search space is a big city. Given this cell size, we have a total of approximately 1.1 mil cells in Istanbul and 240k cells in Gowalla.

We run experiments in two scenarios: *noisy* and *noise-free*. In the *noise-free* scenario, a DPT is applied to the data (which, by Definition 1, is also a RPT) and the attacker observes the true answers to \mathcal{R} . In the *noisy* scenario, we assume that \mathcal{R} is answered by a third party (external) service. For privacy reasons, the service randomly returns false answers to $p\%$ of the queries, whereas the remaining $(100 - p)\%$ is answered truthfully.

We repeat each experiment 100 times by changing the set of known samples and the target, and report the average results.

4.2 Evaluation Metrics

Following the inputs and outputs of Algorithm 1, we use the following transcript to denote the attack: $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}}, K, E) = U'$, where \mathcal{A} denotes the attack, $(U, \mathcal{R}_{\mathcal{D}}, K, E)$ denote the four input parameters as described in Algorithm 1, and U' is the output, i.e., the portion of the search space that the attack claims the target record r_E is located in. We quantify accuracy as follows:

$$\text{Accuracy} = \Pr(r_E \in U' | \mathcal{A}(U, \mathcal{R}_{\mathcal{D}}, K, E) = U')$$

The probability boils down to the empirical ratio of trials where the attack was right in predicting that r_E was located in portion U' divided by the total number of trials. We underline that the accuracy of our attack is always 100 percent when there is no added noise, i.e., the data transformation is relation preserving.

Our second metric is *precision*. An attack that simply outputs $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}}, K, E) = U$ without doing anything achieves 100 percent accuracy, but it cannot be considered successful since it is very imprecise. The success of the attack lies in its ability to identify a *small* portion $U' \subseteq U$ that r_E is located in. This is captured by the *precision* metric we define below. Let $\text{vol}(\cdot)$ denote the volume of a given n -dimensional region. Given $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}}, K, E) = U'$:

$$\text{Precision} = \frac{\text{vol}(U - U')}{\text{vol}(U)}$$

In a uniform-cell based implementation, precision can be calculated simply by dividing the number of pruned cells (i.e., those in $U - U'$) by the total number of cells in the universe U .

Our third metric is *RMSE*, the root mean square error of estimation. Since the output of the attack is U' , the attacker estimates that the target record is randomly assigned to a point in a cell in U' . To measure the error in this estimation, given the cells $C_i \in U'$ and the maximum distance allowed by the boundaries of the data space $\rho(U)$, we calculate:

$$\text{RMSE} = \sqrt{\frac{\sum_{C_i \in U'} \left(\frac{\delta(\mu(C_i), r_E)}{\rho(U)} \right)^2}{\# \text{ of cells in } U'}}$$

where $\mu(C_i)$ denotes the center of cell C_i . Note that the actual distance $\delta(\mu(C_i), r_E)$ is normalized by $\rho(U)$ to ensure we do not face granularity errors and results remain comparable across multiple datasets.

4.3 Results and Discussion

The most interesting aspect of our attack is how precision changes with respect to the number of known samples. With this, we can directly infer how many samples would be needed to locate a target record. We run this experiment in the noise-free scenario, and the noisy scenario with varying levels of perturbation ($p\%$). We graph the results in Fig. 3. In the noise-free scenario, the attack achieves 96 percent precision with 4 or more known samples. Even with 2 samples, the attack achieves 69 percent and 76 percent precision on the Istanbul and Gowalla datasets respectively. Considering that it is easy to obtain a sample of 2-4 records (e.g., check-ins of the attacker himself and 1-3 acquaintances) we can deduce that our attack is very feasible in practice.

Using Fig. 3, we also compare the precision between noisy and noise-free scenarios. For the noisy scenario, (i) we only consider the precision of those records that are correctly located, i.e., $r_E \in U'$ given that $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}}, K, E) = U'$, and (ii) we choose an average voting threshold of 0.6 (the choice of this value will be justified later when we discuss Fig. 6). We make several observations: (i) Precision is higher in the noise-free scenario than in the noisy scenarios. However, their difference is only 10 percent, which implies that the attack is resilient against noise. (ii) When we increase the amount of noise (i.e., $p\%$), precision drops, but only slightly. (iii) More known samples yields higher precision. Interestingly, in the noisy scenarios, it appears that precision is “capped” at some level (93-94 percent) and cannot increase further. We observed that this is caused by our voting mechanism: A voting threshold of 0.6 means that the majority of the known samples must agree to prune a cell before that cell can be pruned. This causes some borderline cells which are easily pruned in the noise-free scenario to remain unpruned in the noisy scenario.

In Fig. 4, we repeat the same experimental setting but report results using the RMSE metric. As expected, higher precision yields lower RMSE. RMSE values are higher in the noisy scenarios than the noise-free scenario, which can again be explained using the selective pruning of our voting mechanism which results in larger unpruned regions. However, even though there is a clear relationship between the percentage of noise and precision (see Fig. 3a), there is no such relationship between percentage of noise and RMSE, i.e., they do not seem correlated. This can be explained in conjunction with the steady decrease in RMSE with more known samples. We observed that pruned/unpruned regions and cells are often adjacent, e.g., in Fig. 2, we do not end up with the bottom right corner and upper left corner simultaneously unpruned. Therefore the unpruned regions (the sole factor affecting RMSE)

1. <http://snap.stanford.edu/data/loc-gowalla.html>

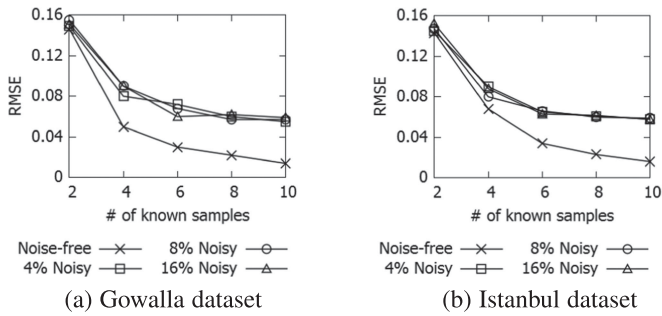


Fig. 4. RMSE in noise-free and noisy scenarios.

are near the target record. If the opposite were true, we would have seen clear increase in RMSE with higher noise.

We comment on the variance of the obtained results by building 95 percent confidence intervals. We observe that variance drops significantly when we have more known samples. For example, with 2 known samples, confidence intervals of precision and RMSE are $74\% \pm 5\%$ and 0.143 ± 0.02 respectively. However, with 6 known samples, the confidence intervals are $96\% \pm 1.7\%$ and 0.034 ± 0.007 respectively. This is explained as follows: When we have few known samples, since these samples are chosen randomly, their locations and relations with the target record have a higher impact on the amount of pruning. In some cases, we can prune as high as 95 percent of the universe, but in others, we can prune only 10 percent. Thus, variance is high. On the other hand, when we have 6 known samples, there is almost always some pairs that lead to effective pruning. As a result, the lower bound on the amount of pruning is high, e.g., we achieve at least 90 percent precision in every experiment. Thus, variance is low.

Next, in Fig. 5, we show the accuracy of our attack with respect to the number of known samples. Interestingly, accuracy drops as we have more samples. This is because of noise. Alg. 1 prunes for every pair of samples, and more samples imply more pruning. However, at the same time, more pruning increases the probability of erroneous pruning (i.e., pruning space that actually contains r_E) due to the existence of noise. This gives a clear trade-off between precision and accuracy: If we are more aggressive by pruning many regions, we increase our precision. However, at the same time, it becomes more likely that the region containing the target record will also be pruned if we are too aggressive. This would decrease our accuracy.

Even though we do not achieve 100 percent accuracy in case of noise, we observe that for those attacks falsely pruning the target point, the corresponding RMSE values are similar to those of the successful attacks. This means unpruned regions stand in close proximity to the target point. We stress that even an unsuccessful attack can disclose the whereabouts of a target.

Taking into account Figs. 3, 4, 5, we describe some guidelines for the attacker. The goal of the attacker should be to first remain accurate in his prediction, and second to make his prediction very

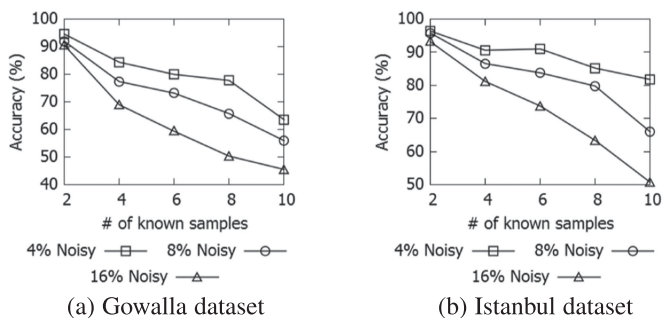


Fig. 5. Accuracy in the noisy scenario.

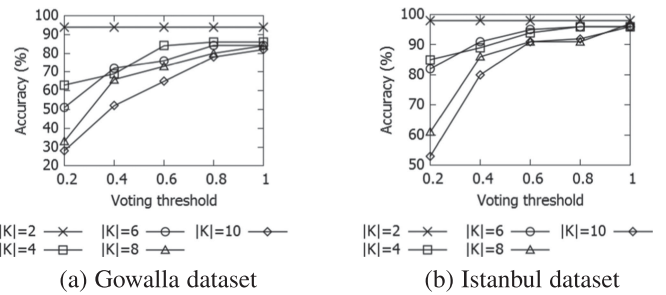


Fig. 6. Effects of the voting threshold on accuracy.

precise. For example, if the attacker very precisely locates his target within 0.01 percent of the universe but the target is not actually located there, this is potentially a bigger problem than having a somewhat poorer precision, e.g., 4 percent, but remaining accurate in the prediction. Therefore we argue that accuracy should not be sacrificed in favor of precision.

One factor that affects accuracy in the noisy scenario is the voting threshold. We fix the noise level to 8 percent and perform experiments with varying number of known samples and thresholds, as illustrated in Fig. 6. We make the following observations: First, as the voting threshold increases, accuracy increases. This is because a higher voting threshold implies more known samples need to agree in order to prune a region, and therefore a single noisy answer is less likely to cause erroneous pruning. Second, as we have more samples, we should have higher voting thresholds in order to remain accurate. This is in parallel with the results and discussion of Fig. 5 - more samples usually cause accuracy to drop, and the voting threshold should be increased to compensate. Finally, we observe that for both datasets, voting thresholds of at least 0.6 or 0.8 are needed to remain roughly 80 percent accurate.

5 CONCLUSION

In this paper, we described and evaluated an attack on a generalized view of distance preserving transformations: relation preserving transformations. Our attack can reach high levels of accuracy and precision with only few known samples. For example, we can locate a target record with over 94 percent precision using only 4 known samples. In addition, we showed that our voting mechanism can effectively combat noise.

We believe that incorporating additional background knowledge such as map information (roads, points of interest, ...) or distribution of data points could enable the attacker further prune regions, thus achieve more precise attacks. Distribution of points could also be used to better evaluate the risk of identification.

Even though formulating such attacks is interesting, one should also study how we can defend against them. Introducing noise to disturb distance relations decreases attack accuracy, but this comes at the cost of utility for distance sensitive data mining operations. Due to exponential time complexity of the attack, one could also consider publishing data of higher dimensionality. However, possible attack improvements such as precomputation of samples and the use of multi-granularity grids still stand as a risk. How effective such improvements would be in practice is an open question. In future work, we plan to study the effectiveness of possible solutions using privacy techniques such as multiplicative perturbation [6], geometric perturbation [10] and aggregation-based methods [14]. We will also study the potential implications of our attack on order-preserving encryption [15].

REFERENCES

- [1] K. Chen and L. Liu, "Geometric data perturbation for privacy preserving outsourced data mining," *Knowl. Inf. Syst.*, vol. 29, no. 3, pp. 657–695, 2011.

- [2] S. R. Oliveira and O. R. Zaiane, "Achieving privacy preservation when sharing data for clustering," in *Proc. Workshop Secure Data Manag.*, 2004, pp. 67–82.
- [3] K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," in *Proc. IEEE Int. Conf. Data Min.*, 2005, pp. 589–592, <https://dl.acm.org/citation.cfm?id=1106402>
- [4] J.-W. Huang, J.-W. Su, and M.-S. Chen, "FISIP: A distance and correlation preserving transformation for privacy preserving data mining," in *Proc. Int. Conf. Technol. Appl. Artifi. Intell.*, 2011, pp. 101–106.
- [5] C. R. Giannella, K. Liu, and H. Kargupta, "Breaching euclidean distance-preserving data perturbation using few known inputs," *Data Knowl. Eng.*, vol. 83, pp. 93–110, 2013.
- [6] K. Liu, C. Giannella, and H. Kargupta, "A survey of attack techniques on privacy-preserving data perturbation methods," in *Proc. Priv.-Preserving Data Min.*, 2008, pp. 359–381.
- [7] K. Liu, C. R. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *Proc. Eur. Conf. Principles Data Min. Knowl. Discovery.*, 2006, pp. 297–308.
- [8] S. Guo and X. Wu, "Deriving private information from arbitrarily projected data," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Min.*, 2007, pp. 84–95.
- [9] E. O. Turgay, T. B. Pedersen, Y. Saygı, E. Savaş, and A. Levi, "Disclosure risks of distance preserving data transformations," in *Proc. Int. Conf. Sci. Stat. Database Manag.*, 2008, pp. 79–94.
- [10] K. Chen, G. Sun, and L. Liu, "Towards attack-resilient geometric data perturbation," in *Proc. SIAM Int. Conf. Data Min.*, 2007, pp. 78–89.
- [11] B. D. Okkalioglu, M. Okkalioglu, M. Koc, and H. Polat, "A survey: Deriving private information from perturbed data," *Artif. Intell. Rev.*, vol. 44, no. 4, pp. 547–569, 2015.
- [12] E. Kaplan, T. B. Pedersen, E. Savaş, and Y. Saygı, "Discovering private trajectories using background information," *Data Knowl. Eng.*, vol. 69, no. 7, pp. 723–736, 2010.
- [13] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2011, pp. 1082–1090.
- [14] C. C. Aggarwal and P. S. Yu, "A condensation approach to privacy preserving data mining," in *Proc. Int. Conf. Extending Database Technol.*, 2004, vol. 4, pp. 183–199.
- [15] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2009, pp. 224–241.