Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering (ESAT)

Security Architecture for Things That Think

Beveiligingsarchitectuur voor Intelligente Objecten

Roel PEETERS

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

June 2012

Security Architecture for Things That Think

Beveiligingsarchitectuur voor Intelligente Objecten

Roel PEETERS

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering

June 2012

D/2012/7515/47
ISBN 978-94-6018-511-3

Greetings, my friend.

We are all interested in the future,

for that is where you and I are going to spend the rest of our lives.

And remember my friend,

future events such as these will affect you in the future.

You are interested in the unknown . . . the mysterious . . . the unexplainable.

That is why you are here.

Plan 9 from Outer Space.

Acknowledgements

In de eerste plaats wil ik Prof. Bart Preneel bedanken voor de geweldige kans om een doctoraat te starten, voor zijn wijze raad en tips in verband met presenteren. Verder ben ik hem ook dankbaar voor zijn steun aan mijn plan om een half jaar onderzoek te gaan doen in Denemarken. I would also like to thank Prof. Ivan Damgård for hosting my research visit and assuming the role of advisor during my stay in Denmark.

I am grateful to Prof. Vincent Rijmen, Prof. Patrick Wambacq, Prof. Frank Stajano and Prof. Hervé Chabanne for serving as jury members; and to Prof. Yves Willems for chairing the jury. Special thanks go to Prof. Frank Stajano for reading this thesis very thoroughly, providing lots of suggestions and correcting my English grammar mistakes.

Ik wil de KU Leuven, het agentschap voor Innovatie door Wetenschap en Technologie (IWT) en Aarhus Universitet bedanken om mijn onderzoek financieel mogelijk te maken.

Thanks to all my co-authors and discussion partners for research related topics. Bedankt Dave, om mijn voorbeeld te zijn. Dankjewel Koen, om me keer op keer te overtuigen dat het onmogelijke toch mogelijk is (onder bepaalde veronderstellingen) en al je hulp om mijn schrijfstijl te verbeteren. Bedankt Fré, voor alle wiskundige achtergrond en je ongebreideld enthousiasme telkens we nog enkele dagen voor een deadline plots met een wiskundig probleem aankwamen. Bedankt Jens[1] om me te overtuigen van de kracht van bewijzen voor protocols en alle discussies. Bedankt Nicky om me te helpen bij het opzetten van een gebruikersevaluatie, het was een zeer leerrijke ervaring. Bedankt Roel, Anthony en Ingrid voor een beetje inzicht in de hardwaregerelateerde kant van protocollen. Thank you Junfeng for answering all questions about hardware specifications. Thank you Dave, Stefaan, Jens and Andreas for proofreading this text.

[1]Thank you Filipe for volunteering your chair during my discussions with Jens.

A big thanks to all my colleagues, without you it would have been boring. Bedankt Elke en Wim voor de gezellige babbels in ’t lab, waarbij de besjestheorie [5] mij zeker zal bijblijven. Bedankt Sebastiaan voor alle zotte plannen die we uiteindelijk toch nooit uitvoerden. Bedankt Stefan en Elmar om jullie liefde voor eten en drinken met mij te delen. Ne dikke merci aan de bende van het zwart hemd en rode das: Karel, Yoni, Stefaan, Dave en Dries voor al het plezier, RAAK. A special thanks to all my office mates, of whom Li is the constant during all the internal moves in the building. Thank you everyone for the nice atmosphere, the BBQ’s, the weekends and the friday beers.

Tusind tak til krypto-gruppen i Århus, min forskning besøg var meget hyggelig! During my stay I learnt a lot and had very interesting discussions with most of you, especially Ivan, Carmit, Jesper, Thomas and Nik. I also learnt that Danes are weird, but a very friendly people once you get to know them. Thanks for all good times everyone and I hope to see you all in Belgium one day for some nice “stoofvlees”[2] with french fries.

Péla is de geweldigste secretaresse bij wie ik altijd terechtkan voor een babbel en allerlei vragen. Dankjewel voor de impulsen aan sociale activiteiten en je enthousiasme. Bedankt Saartje, voor het beheren van de projecten die mij toelieten om ook na het verlopen van mijn beurs te blijven op COSIC. Tusind tak til Sara og Dorthe på Århus Universitet for hjælpen. Dankjewel Elsy, dankzij jou kregen we snel al onze gemaakte kosten terugbetaald. Graag bedank ik ook Elvira voor alles en om me nooit mijn eerste dag op COSIC te laten vergeten.

Het afgelopen jaar was voor mij een emotionele achtbaan. Mama, ik ben er zeker van dat je enerzijds ongelofelijk trots zou zijn op mijn doctoraat. Daarom draag ik het in blije herinnering aan jou op. Anderzijds zou je minstens even gelukkig zijn dat ik nu met Birgit de liefde ben tegengekomen. Ik wil alle vrienden en familie bedanken voor hun steun in de moeilijkere momenten en de gedeelde vreugde op andere momenten, in het bijzonder papa, Angela en Karen. Angela, grazie mille per tutti i bei momenti! Dankjewel Birgit voor je steun, begrip en je liefde voor mij.

Roel Peeters

Juni 2012

PS: If you don’t like to go into the technical details, please skip the next 153 pages and go right ahead to page 142 ☺.

[2] According to Dorthe, who is also a language expert, this is related to the danish “stole".

Abstract

The observation that people already carry lots of personal devices (e.g., a smart phone, an electronic identity card, an access badge, an electronic car key, a laptop, ...), serves as starting point for this thesis. Furthermore, with the arrival of smart objects, the number of things that think one carries is expected to grow. Sensors will be built into clothing and attached to the body to monitor our health. It is clear that these devices need to be protected. However, due to the vast amount of devices involved, the traditional approach of protecting each device on its own, results in a usability nightmare.

We investigate how to tap into the potential that arises from cooperation between these devices. This is done by deploying threshold cryptography on personal devices. Threshold cryptography has the benefit of increased overall security, since an adversary can compromise a number (up to the threshold number) of devices without gaining any advantage towards breaking the overall security. Furthermore, the end-user does not need to carry all his personal devices, any subset of size at least the threshold number is sufficient to make use of the threshold security system.

We propose technical solutions to tackle some of the practical issues related to this approach, paving the road for real world implementations. First, we show how one can include devices that do not have the necessary (secure) storage capabilities needed to store shares (e.g., car keys) in our threshold scheme. Second, we investigate how the end-user can add or remove devices from his set of personal devices used in this threshold scheme. Finally, in order to get user acceptance, the (location) privacy of consumers should not be disregarded. Towards this goal we examine how to achieve private and secure device authentication over an open channel. This is done specifically for RFID tags, which are the least powerful devices that can be included in our threshold system. Hence, the location of the end-user can be kept private, while all communication between his personal devices, that arises from our threshold solution, goes over an open channel.

Beknopte samenvatting

Het uitgangspunt van dit proefschrift is de observatie dat veel mensen al een behoorlijk aantal persoonlijke toestellen met zich meedragen, bijvoorbeeld een smartphone, een elektronische identiteitskaart, een toegangsbadge, een elektronische autosleutel of een laptop. Door de technologische vooruitgang duiken allerlei nieuwe intelligente objecten op. Dit aantal wordt verwacht nog te groeien met allerlei sensoren die zowel op kledij als op het lichaam worden aangebracht om onze gezondheid op te volgen. Het is belangrijk dat deze intelligente objecten afdoende beschermd worden. Door het grote aantal toestellen dreigt de traditionele aanpak om elk toestel op zich te beschermen, al vlug te ontaarden in een nachtmerrie voor de gebruiker.

We onderzoeken hoe we kunnen profiteren van de samenwerking tussen deze intelligente objecten. Hiertoe gebruiken we drempelcryptografie op de persoonlijke toestellen. Door drempelbeveiliging te gebruiken kan een aantal (kleiner dan de drempelwaarde) toestellen gestolen worden zonder dat de beoogde bescherming in het gedrang komt. Bovendien laat dit ook toe dat de gebruiker het beveiligingssysteem kan gebruiken zolang hij minstens de drempelwaarde aan toestellen bij zich heeft, met andere woorden niet alle toestellen moeten aanwezig zijn.

We bieden technische oplossingen voor enkele van de praktische problemen gerelateerd aan deze aanpak, waardoor we een stap dichter komen bij de realisatie van drempelbeveiliging op intelligente persoonlijke objecten. In de eerste plaats tonen we hoe men toestellen die niet over extra (veilige) opslag beschikken, zoals bijvoorbeeld een elektronische autosleutel, toch kunnen toevoegen aan het drempelbeveiligingssysteem. Ten tweede gaan we ook na hoe de gebruiker toestellen kan toevoegen of verwijderen uit de groep van persoonlijke toestellen die hij gebruikt voor zijn drempelbeveiligingssyteem. Ten slotte mogen we de persoonlijke levenssfeer van de gebruikers niet negeren. Er moet voor gezorgd worden dat alle communicatie tussen de persoonlijke toestellen de locatie van de gebruiker niet prijsgeeft. Om dit doel te bereiken gaan we na hoe we een veilige en private authenticatie kunnen bekomen over een open kanaal. We bekijken dit specifiek vanuit het standpunt van RFID-tags, de minst krachtige apparaten die kunnen deelnemen in het voorgestelde drempelbeveiligingssysteem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Things That Think

Things That Think or *Smart Devices* are networked devices with computational capabilities. Typically these devices are mobile, hence constrained by the capacity of their batteries or by the amount of energy that can be drawn from the environment. Mobility also means that communication between these devices is wireless.

In the context of sensing, one refers to *Wireless Sensor Networks* (WSN) [49, 153]. These networks consist of autonomous sensors and can be deployed to monitor a wide range of physical phenomena such as light, noise, temperature, energy consumption, traffic, etcetera. The term *Smart Dust* [98, 165] is used for sensor networks that consist only of very small sensors (dimensions in the order of millimetres or smaller). Sensor networks can also be deployed to monitor the health of an individual, in this context the term *Body Area Network* (BAN) [39, 159] is used. Sensors in the human body (or embedded into clothing) will allow to constantly monitor a patient's health status. This is particularly beneficial for people with a high risk factor for heart attacks and patients who suffer from chronic diseases such as diabetes and asthma. Nowadays, these sensor networks are becoming bi-directional, besides from sensing the environment one can also influence it by directing actuators.

In the *Internet of Things* (IoT) [7, 93], each thing (object) has a unique identifier and is interconnected in an internet-like structure. *Radio Frequency IDentification* (RFID) is often considered to be a prerequisite for the Internet of Things, as RFID tags allow to transform everyday objects into smart objects.

The processing of information, i.e. computing, has become ever more integrated into everyday objects and the environment. Consumers engage many things that think and may not even be aware of doing so. This paradigm is also referred to as *Ubiquitous Computing* [167, 168], *Pervasive Computing* [79] and *Ambient Intelligence* [1]. A popular scenario is the one of the smart refrigerator that is aware of its contents. The end-user is thus able to plan menus from the available food, and will be warned when goods are close to their expiry date. This would mean that every food item needs to be labelled in an appropriate way. In the same way, smart shelving can be deployed in shops, alerting the shop owner when items need to be restocked on the shelves or when to remove goods that have gone bad. Similarly, clothing could contain appropriate labels with washing instructions, making smart washing machines possible. Although these scenarios seem futuristic, ubiquitous computing is already around us in our daily lives in many ways. Think for instance of smart phones, RFID tags, GPS systems, the possibilities to stream digital audio and video across devices, home automation and interactive displays. Some car models allow to identify the driver (e.g., through a specific key) and adjust the seat and mirrors according to his preferences.

At the end of 2008, the European Commission published a roadmap for the future of the Internet of Things [93]. This report states current trends in technology, applications and society. An outlook for the future, towards 2020, is given. It is projected that RFID tags will be all around us, ubiquitously integrated with wireless sensor networks by 2015. Real smart objects and truly ubiquitous computing are expected in the period between 2015 and 2020. This document also lists privacy and security as possible barriers towards widespread adoption. In other words, there is a need for technical solutions to address the privacy and security needs of the end-users.

1.1.1 Personal Devices

This thesis primarily focusses on personal devices, i.e. things that think that end-users carry on a regular basis. Nowadays, one typically always carries a phone, a car key and a wallet, containing one or more smart cards. On top of that, many people carry some sort of access token, needed to get access to restricted areas in the company they work for. Other examples of personal devices include a passport, a laptop, a digital camera, an mp3-player, a GPS receiver, a digital watch, a tablet, a heart rate monitor. Future BANs and things that think, which are woven into clothing, are also considered to be personal devices.

We develop a security architecture for things that think that aims to protect both the individual devices as well as the data these host. First, the individual devices will be protected against stealing or accidental loss, by making their functionality unavailable when the adversary only has access to a single device. More concretely, this means that an adversary that obtained your car key has not automatically access to your car, an access token alone is not sufficient for accessing restricted zones, a mobile phone will not be able to make calls that will be charged to the rightful owner. Second, also the data on these devices need to be protected, by deploying an encryption scheme.

For many years, sensitive information has been protected by a corporate shield set up by administrators maintaining firewalls, intrusion detection systems, anti-virus programs, etcetera.. However, this responsibility is shifting towards the end user, who also retrieves sensitive information through his mobile devices such as laptops and smart phones. By accessing the information, it gets copied onto the mobile device where it remains unprotected. For example, anyone who has physical access to an unprotected smart phone can:

- read stored emails, possibly containing confidential information;
- know your meeting schedule;
- get a list of your friends or business contacts and the corresponding phone numbers, addresses, pictures;
- read your stored text messages;
- have a look at your personal pictures.

From a corporate point of view, mobile devices also have a big security impact. Each year, the Computer Security Institute gathers data, on a voluntary basis, from security practitioners in the US who report the experienced security related incidents in their companies. The 2011 survey [41] shows that 34% of their respondents experienced theft or loss of laptops or mobile hardware. This theft or loss of mobile devices resulted in theft or unauthorised access to: (1) private identifiable information or private health information (claimed by 5% of the respondents); (2) intellectual property (again 5%).

Both security objectives rely on how one authenticates the consumer towards his personal devices.

1.2 Protecting Things That Think

User authentication is tradionally based on one of three things:

- Something you are, e.g., your fingerprint.
- Something you have, e.g., your access card.
- Something you know, e.g., your password.

Today, passwords and PINs are widely used. However, with a multitude of accounts and devices to secure, passwords become ever more unmanageable for consumers. Passwords should be different for every application, hard to guess and never written down. However, reliability and convenience are often more important to consumers than security. For this reason, one should try to steer away from all related trade-offs between memory and security [4]. Different solutions for passwords were proposed in the context of online application, ranging from password managers to Single Sign On applications. Apart from the issues related to these solutions, it does not help in protecting offline applications. To rid the end-user of all passwords, Stajano [156] proposed the use of an additional device called *Pico.* Herley and van Oorschot [83] argued that research should, instead of focussing on replacing all passwords, acknowledge that passwords will be around for quite some time and that these are the best solution for many cases. To deal with an untrusted computer (no key-logging) and to provide transaction integrity (no phishing or session-hijacking), Mannan and van Oorschot [114] came up with the MP-Auth protocol. Instead of providing his password on the untrusted host, the user inputs his password on a trusted personal device, e.g. mobile phone. Pashalidis [125] came up with another approach to deal with untrusted computers, with the a service named *Keep Your Password Secret* that allows the end-user to logon to web services by using only one time passwords.

Some applications also use two-factor authentication. The most prominent examples can be found in payments systems and online banking, for which both a bank card and a PIN are needed. Another example that is worth mentioning is the optional two-factor authentication process for Google applications [75], that requires a mobile phone and a password.

With the multitude of personal devices, multi-factor authentication is possible. Instead of securing each device separately, we will make use of their ability to interconnect and offer a global solution. This general solution should also be able to cope with situations where some devices are absent, to avoid hurting the reliability. Threshold cryptography [50] can be used to provide this solution. Stajano [156] also suggests this approach as a means to secure the Pico. In this context, these devices are referred to as *Picosiblings.*

1.3 Summary of Contributions

The observation that people already carry lots of personal devices, serves as starting point for this thesis. Furthermore, with the arrival of smart objects, the number of things that think one carries is expected to grow. Instead of protecting each device on its own, we investigate how to tap into the potential that arises from cooperation between these devices. This is done by deploying threshold cryptography on personal devices. We propose technical solutions to tackle some of the practical issues related to this approach, paving the road for real world implementations. In order to get user acceptance, the (location) privacy of consumers should not be disregarded. Towards this goal we examine how to achieve private and secure device authentication over an open channel. This is done specifically for RFID tags, which are the least powerful devices that can be included in our threshold system. Hence, the location of the end-user can be kept private, while all communication between his personal devices, that arises from our threshold solution, goes over an open channel. More specifically our contributions are:

- A new way of securely storing shares, i.e. in a protected format, such that these shares can be made public. This enables us to include devices that do not have the necessary (secure) storage capabilities needed to store shares. We propose a *Verifiable Secret Sharing* (VSS) and a *Distributed Key Generation*(DKG) protocol to set up these protected shares and show how to transform some existing cryptosystems and signature schemes [151].
- We discuss possible approaches to deal with absent devices while redistributing the shared secret (resharing) among the personal devices.
- A study of how to authorise resharing, possibly altering the group of personal devices. We propose a protocol that allows us to authorise a request for resharing [128] and evaluate the usability of this protocol [129].
- An existing RFID privacy model is extended to allow for private authentication towards multiple independent readers, a stepping stone for device-to-device authentication. This extended model also captures recently proposed insider attacks. We also provide a definition for mutual authentication in this context.
- An efficient, secure and private RFID protocol is proposed. We prove both the security and privacy properties of this protocol in our extended RFID privacy model. Furthermore, we provide a mutual authentication protocol that is efficient and privacy-preserving.

We worked on other research publications that have not been included in this doctoral thesis:

- In [131, 132, 133], we combine the concept of threshold cryptography on personal devices with cryptographic distance bounding protocols, in order to achieve a resilient proximity-based physical access control system.
- In [111], we design a lightweight authentication protocol for RFID tags, by using Physical Unclonable Functions (PUF). We propose a new construction to deal with the noisy output of the PUF, the reverse fuzzy extractor. This construction allows us to shift some of the computational effort towards the verifier.
- In [85], we model private RFID yoking proofs. A private RFID yoking proof attests that two tags were scanned simultaniously, while the identities of the RFID tags can only be learnt by the verifier. We evaluate an existing protocol and propose a new provable secure and private protocol to generate such a proof.
- In [130], we study cross-context delegation through identity federation. More specifically we look at how one issues credentials for certain privileges to another person (mandate), accepts a mandate, revokes the issued mandate and passes on a mandate. We approach delegation from the perspective of individuals as well as companies.

1.4 Thesis Outline

Chapter 1 provides a general introduction and introduces the necessary mathematical background.

Chapter 2 gives an introduction to threshold cryptography. Furthermore we discuss the benefits of deploying threshold cryptography on personal devices. The communication and adversarial model is introduced for the context of threshold cryptography on personal devices.

Chapter 3 proposes a threshold scheme that can also include less powerful devices, making the overall system more reliable. The shares are blinded and publicly available. By also introducing public verifiability in the setup, not all devices need to be present at that time.

Chapter 4 examines how to transform a threshold system into a proactive threshold system by means of resharing. Resharing also allows one to change the set of personal devices and the threshold number. This chapter takes into account that not all devices might be present at the time of resharing. Furthermore we examine how to involve the user, especially to authorise changes in the set of personal devices.

Chapter 5 deals with private authentication in the context of RFID tags. To ensure that the end user cannot be tracked easily by all the communication between his personal devices and the environment, an important building block is a protocol that enables private authentication over an open channel. More specifically, we look into this from the point of view of the least powerful devices, namely RFID tags. Since these tags do not have any user interface, these respond to any valid query. We show how we can use this private authentication in our threshold scheme in order to keep the end-user's location private.

Chapter 6 concludes this thesis and gives directions for future research.

1.5 Mathematical Background

1.5.1 Elliptic Curves

Elliptic curves over a finite field are an alternative to directly using finite fields in cryptography. An example of an elliptic curve is given in Fig. 1.1. For a comprehensive overview of elliptic curve cryptography we refer the reader to [9].

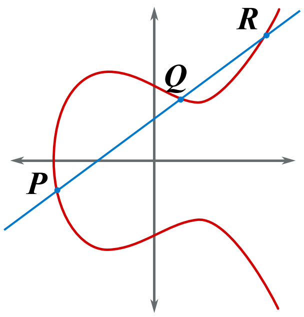


Figure 1.1: Example of an Elliptic Curve.

An elliptic curve $\mathbb{E}$ can be specified using the Weierstrass equation:

$$y^2 = x^3 + ax + b,$$

where all variables and constants are elements of the finite field. All solutions (x, y) to the above equation are (rational) points on the elliptic curve, i.e.

$(x, y) \in \mathbb{E}$. Besides these points, the point at infinity O is added to the set of points. This point will serve as the identity element. Curves are typically defined over $\mathbb{F}_p$ (prime curves) or over $\mathbb{F}_{2^n}$ (binary curves).

Let $\mathbb{E}_\ell$ be an elliptic curve with prime order ℓ over $\mathbb{F}_p$. Points on the elliptic curve are denoted by capital letters, while scalars are denoted by lower case letters. We make use of additive notation. A point Q on the elliptic curve can be represented as $\{q_x, q_y\}$ with $q_x, q_y \in [0 \ldots p-1]$.

The group law (point addition) is defined by taking two elliptic curve points P, Q and constructing a straight line through these points. There is always a third intersection point $R = (r_x, r_y)$ on the curve. The result of the addition is $-R = (r_x, -r_y)$. Note that $P + Q + R = O$. In case $P = Q$ (point doubling) the tangent line to the curve at P is considered to determine R.

We now consider scalar multiplication, i.e. given an $a \in \mathbb{Z}_\ell$ and a point $P \in \mathbb{E}_\ell$, compute $aP = \sum_{i=1}^{a} P$. Scalar multiplication can be computed efficiently by the double-and-add algorithm, which requires at most $\log_2 a$ point additions and doublings.

The $\texttt{xcoord}(\cdot)$ function is the ECDSA conversion function [29], which is almost-invertible. The function $\texttt{xcoord}(Q)$ maps a point Q to the scalar $q_x \mod \ell$.

1.5.2 Pairings

Pairings are bilinear maps and are usually defined over elliptic curve groups. A comprehensive overview of pairings for cryptographers is given by Galbraith *et al.* [65].

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of order ℓ and let $\hat{e}$ be a non-degenerate bilinear pairing

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T \, .$$

A pairing is non-degenerate if for each element P in $\mathbb{G}_1$ there is a Q in $\mathbb{G}_2$ such that $\hat{e}(P, Q) \neq 1$ and vice versa for each element Q in $\mathbb{G}_2$. A pairing is bilinear if $\hat{e}(P + P', Q) = \hat{e}(P, Q)\hat{e}(P', Q)$, thus $\hat{e}(aP, Q) = \hat{e}(P, Q)^a$ with $a \in \mathbb{Z}_\ell$, and vice versa for elements in $\mathbb{G}_2$. We will use multiplicative notation for $\mathbb{G}_T$ and additive notation for $\mathbb{G}_1$ and $\mathbb{G}_2$.

1.5.3 Number-Theoretic Assumptions

Security in public-key cryptography usually depends on the intractability of some number-theoretic problem. This means that a protocol or scheme is secure under the condition that certain number-theoretic assumptions hold. We review some relevant assumptions and refer the reader to Mao's [115] book on modern cryptography and to the paper of Smart and Vercauteren [152] on asymmetric pairings for more details.

The security parameter measures the input size of the problem. Both the resource requirements of the cryptographic algorithm or protocol as well as the adversary's probability of breaking security are expressed in terms of the security parameter. The security parameter is usually expressed in unary representation, i.e. 1^k for a k-long string of bits.

A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is called 'polynomial' in the security parameter $k \in \mathbb{Z}$ if $f(k) = O(k^n)$, with $n \in \mathbb{N}$. It is called 'negligible' if, for every $c \in \mathbb{N}$ there exists an integer k_c such that $f(k) \leq k^{-c}$ for all $k > k_c$.

Discrete Logarithm. Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ and let A be a given arbitrary element of $\mathbb{G}_\ell$. The discrete logarithm (DL) problem is to find the unique integer $a \in \mathbb{Z}_\ell$ such that $A = aP$. The DL assumption states that it is computationally hard to solve the DL problem.

One More Discrete Logarithm. The one more discrete logarithm (OMDL) problem was introduced by Bellare *et al.* [18]. Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ. Let $\mathcal{O}_1$ be an oracle that returns random elements $A_i = a_iP$ of $\mathbb{G}_\ell$. Let $\mathcal{O}_2(\cdot)$ be an oracle that returns the discrete logarithm of a given input base P. The OMDL problem is to return the discrete logarithms for each of the elements obtained from the m queries to $\mathcal{O}_1$, while making strictly less than m queries to $\mathcal{O}_2(\cdot)$.

x-Logarithm. Brown and Gjøsteen [30] introduced the x-Logarithm (XL) problem: given an elliptic curve point, determine whether its discrete logarithm is congruent to the x-coordinate of an elliptic curve point. The XL assumption states that it is computationally hard to solve the XL problem. Brown and Gjøsteen also provided some evidence that the XL problem is almost as hard as the DDH problem.

Diffie-Hellman. Let P be a generator of a group $\mathbb{G}_\ell$ of order ℓ and let aP, bP be two given arbitrary elements of $\mathbb{G}_\ell$, with $a, b \in \mathbb{Z}_\ell$. The computational Diffie-Hellman (CDH) problem is, given P, aP and bP, to find abP. The tuple $\langle P, aP, bP, abP\rangle$ is called a Diffie-Hellman tuple. Given a fourth element $cP \in \mathbb{G}_\ell$, the decisional Diffie-Hellman (DDH) problem is to determine if $\langle P, aP, bP, cP\rangle$ is a valid Diffie-Hellman tuple or not. The DDH assumption states that it is computationally hard to solve the DDH problem.

Obviously, if one can solve the DL problem then one can also solve the CDH problem. The opposite does not necessarily hold and, therefore, the CDH assumption is said to be a stronger assumption than the DL assumption.

A divisional variant of the DDH problem [12], which is considered to be equivalent, is to determine if $\langle P, aP, cP, abP\rangle$ is a valid DH tuple or not, i.e., if $c = b$

Oracle Diffie-Hellman. Abdalla *et al.* [2] introduced the ODH assumption:

Definition 1. *Oracle Diffie-Hellman (ODH) Given $A = aP, B = bP$, a function H and an adversary $\mathcal{A}$, consider the following experiments:*

Experiment $\mathbf{Exp}^{odh}_{E,H,\mathcal{A}}$ *:*

- $\mathcal{O}_1(Z) = H(bZ)$ *for* $Z \neq \pm A$
- $\mathcal{O}_2() = H(C)$
- $g = \mathcal{A}^{\mathcal{O}_1(\cdot),\mathcal{O}_2()}(A, B)$
- *Return* g

The value C is equal to abP for the $\mathbf{Exp}^{odh-real}_{E,H,\mathcal{A}}$ experiment, chosen at random in $\mathbb{G}_\ell$ for the $\mathbf{Exp}^{odh-random}_{E,H,\mathcal{A}}$ experiment.

The advantage of $\mathcal{A}$ violating the ODH assumption is defined as:

$$\left|\Pr\left[\mathbf{Exp}^{odh-real}_{E,H,\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{odh-random}_{E,H,\mathcal{A}} = 1\right]\right| .$$

The ODH assumption consists of the plain DDH assumption combined with an additional assumption on the function $H(\cdot)$. The idea is to give the adversary access to an oracle $\mathcal{O}_1$ that computes bZ, without giving the adversary the ability to compute bA, which can then be compared with C. To achieve this one restricts the oracle to $Z \neq \pm A$, and moreover, only $H(bZ)$ instead of bZ is released, to prevent the adversary from exploiting the self-reducibility of the DL problem[1].

The crucial property that is required for $H(\cdot)$ is one-wayness. In Chapter 5, we use a one-way function based on the DL assumption. We define the function $H(Z) := \texttt{xcoord}(Z)P$.

Theorem 1. *The function $H(\cdot)$ is a one-way function under the DL assumption.*

Co-Bilinear Diffie-Hellman (coBDH). For asymmetric pairings, i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$, where there is no known efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ the following problem can be defined. The coBDH-2 problem is defined as given $P \in \mathbb{G}_1$ and $Q, aQ, bQ \in \mathbb{G}_2$, find $\hat{e}(P, Q)^{ab}$. We denote the decisional variant as coDBDH-2. A divisional variant of the coDBDH-2 problem is to determine whether $\langle P, Q, aQ, abQ, g^c \rangle$ is a valid coBDH-2 tuple.

Inversion Problems. Galbraith et al. [64] studied several inversion problems for pairings. They concluded that these problems are hard enough to rely upon. The most intuitive argument is that if one can solve a particular pairing inversion in polynomial time then one can also solve a related Diffie-Hellman problem in one of the domains or the co-domain.

The following problems were defined:

- The **Fixed Argument Pairing Inversion 1 (FAPI-1)** problem is: Given $P_1 \in \mathbb{G}_1$ and $z \in \mathbb{G}_T$, find $P_2 \in \mathbb{G}_2$ such that $\hat{e}(P_1, P_2) = z$.
- The **Fixed Argument Pairing Inversion 2 (FAPI-2)** problem is: Given $P_2 \in \mathbb{G}_2$ and $z \in \mathbb{G}_T$, find $P_1 \in \mathbb{G}_1$ such that $\hat{e}(P_1, P_2) = z$.
- The **Generalised Pairing Inversion (GPI)** problem is: Given a value $z \in \mathbb{G}_T$, find a $P_1 \in \mathbb{G}_1$ and a $P_2 \in \mathbb{G}_2$ such that $\hat{e}(P_1, P_2) = z$.

Strong RSA. The RSA problem is to find, given an RSA public key (e, N) and a ciphertext $c = m^e \mod N$, the corresponding plaintext m. The public RSA key consist of a modulus N, which is the product of two large primes p and q, and the exponent $2 < e < N$, which is coprime to $\varphi(N) = (p - 1)(q - 1)$. The strong RSA assumption was introduced by Niko Barić and Pfitzmann [13] and Fujisaki and Okamoto [62].This assumption states that it is hard to solve the RSA problem, even when one can choose the public exponent e.

[1]The adversary can set $Z = rA$ for a known r and compute $r^{-1}(bZ) = bA$.

1.5.4 Cryptographic Primitives

Commitment Scheme. A cryptographic commitment scheme allows to commit to a value without revealing it. Moreover, once committed to this value, one cannot open the commitment to any other value. The first property is the hiding property, the second is the binding. Both properties can be computational or perfect. The property is said to be computational when it holds in presence of adversaries that only have finite amounts of computational power, perfect does not put this limitation on the adversary. A commitment scheme can never be perfect hiding and perfect binding.

There exist schemes that are computational hiding and perfect binding, e.g., Feldmann commitments [54]. For this specific scheme, one commits to a value a by hiding it away in the exponent of a generator g of a group in which the DL assumption holds: $c = g^a$. The DL assumption does not hold for adversaries with unlimited computational power: the scheme is not perfectly hiding. However, even this adversary cannot produce another value a' for which it holds that $c = g^{a'}$: the scheme is perfectly binding.

Vice versa, perfect hiding and computational binding commitments exist, e.g., Pedersen commitments [127]. To commit to a value a, one first chooses a random value b and computes the commitment as follows: $c = g^a h^b$, for g and h generators of the same group in which the DL assumption holds and for which the discrete logarithm of g base h, and vice versa, is unknown. An unbounded adversary can compute the discrete logarithm of $g = h^i$ base h and produce any pair (a', b') as long as the equation $a'i + b' = ai + b$ holds: the scheme is only computationally binding. However, even this adversary cannot produce the value a, given a commitment c: the scheme is perfectly hiding.

Hash Function. A cryptographic hash function $H(\cdot)$ is a one-way function, that takes an arbitrary block of data, the message m, and returns a fixed-size bit string, the hash value $h = H(m)$, such that changes to the data will (with very high probability) result in changes to the hash value. A cryptographic hash function has the following properties:

- **Pre-image resistance**: Given a hash value h, it is hard to find any message m, such that $h = H(m)$;
- **Second pre-image resistance**: Given a message m_1, it is hard to find another message $m_2 \neq m_1$, such that $H(m_1) = H(m_2)$;
- **Collision resistance**: It is hard to find two different messages m_1 and m_2, such that $H(m_1) = H(m_2)$.

Cryptographic hash functions can be used for protecting the integrity of the message.

MAC Algorithm. Message Authentication Code (MAC) algorithms are keyed hash functions. The input is a secret key and an arbitrary-length message, the output is the fixed-length MAC value. The MAC value protects both a message's integrity as well as its authenticity, by allowing the verifier (who also possesses the secret key) to detect any changes to the message.

Chapter 2

Threshold Cryptography

Threshold cryptography can be used to secure data and control access by sharing a private cryptographic key over multiple personal devices. This means that a minimum number of these devices, the threshold number $t + 1$, need to be present to use the key. Threshold cryptography has several benefits: (1) increased overall security, since an adversary can compromise up to t devices without gaining any knowledge on the private key; (2) higher resilience, since any subset of $t + 1$ devices is sufficient for decryption and/or authorised access.

2.1 Introduction

Today, there are only a few real-world applications of threshold cryptography, for example the root certification key for MasterCard/VISA's Secure Electronic Transaction (SET) is a 3-out-of-5 shared RSA key. Blom's scheme [23] is a symmetric threshold key exchange protocol, that is used in the HDCP protocol[1], intended to protect copyright of high definition content between the source, i.e. Blu-ray Disc player, and the receiver, e.g. HD television. The most recent example is the fair auction of sugar beets between farmers and the only sugar producing company in Denmark [24].

The idea of "Threshold Things That Think" was already put forward by Desmedt [51]. But only recently, personal devices are becoming powerful

[1]In Version 1.1, the intended protection could be circumvented by a conspiracy attack: obtaining the keys of at least 40 devices and reconstructing the secret symmetrical master matrix that was used to compute them [44]. Version 1.3 is also reported to be broken [110].

and interconnected enough to enable the deployment of threshold cryptography for securing personal devices.

This chapter first gives a general overview on how to setup a threshold cryptography scheme. Next we will discuss how to deploy threshold cryptography specifically for personal devices.

2.2 Threshold Cryptography

The aim of threshold cryptography is to protect a private key by sharing it amongst a number of entities in such a way that each subset of minimal size, namely the threshold number $t+1$, can use the key. No information about the key can be learnt from t or less shares. For cryptographic operations that use the private key, such as generating signatures or decrypting ciphertexts, the secret key is not reconstructed. Instead, each player contributes a partial signature or partial decryption, for which $t+1$ of these can be combined into a signature or a decrypted plaintext.

Threshold cryptography typically involves routines related to setting up the group, encryption and signatures. We define the following set of routines (threshold routines are indicated with the prefix **T**):

Pre-setup

- **Init:** Initialise the system parameters.
- **KeyGen:** Generate key material for a device.

Setup

- **ConstructGroup:** Given a set of n devices and their public keys, create and share a key pair for the group with a subset of the devices.

The setup of a threshold scheme involves either a trusted dealer or a Distributed Key Generation (DKG) protocol. The trusted dealer simply runs an instance of a secret sharing protocol. In a DKG protocol a group of entities cooperate to jointly generate a key pair and obtain shares of the private key. In this way, no single party will know the value of the shared secret. Both secret sharing and DKG are explained in full detail in the next section.

The shares, constructed during setup, can then be used to sign or decrypt on behalf of the group.

Signatures

- **T-Sign:** At least $t + 1$ devices collaborate to generate a signature on a message that is verifiable under the group's public key.
- **Verify:** Using the group's public key a signature is verified.

Encryption

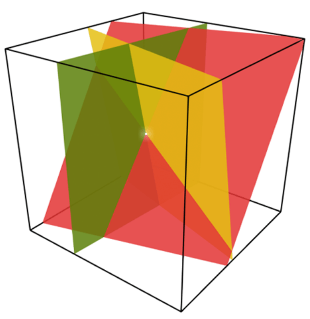
- **Encrypt:** Encrypt a message under the group's public key.
- **T-Decrypt:** At least $t+1$ devices collaborate to decrypt a given ciphertext that was encrypted under the group's public key.

2.3 From Secret Sharing to Distributed Key Generation

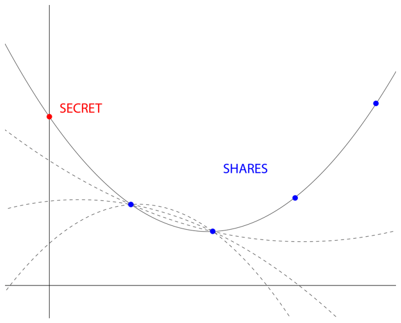
2.3.1 Secret Sharing

Blakley [21] and Shamir [148][2] both proposed a $t + 1$-out-of-n-secret sharing scheme. The total number of shares is n and $t + 1$ is the threshold number, as this is the minimal number of shares needed to reconstruct the secret. In Blakley's secret sharing scheme, for which a conceptual drawing is given in Fig. 2.1(a), each share is a hyperplane of dimension t that contains the point representing the secret. The intersection of any $t + 1$ of these hyperplanes uniquely defines the secret. In Shamir's secret sharing scheme, shares are points on a polynomial of degree t, for which the secret is defined as the free term of this polynomial. Any $t + 1$ of these points uniquely define this polynomial and hence the secret. Figure 2.1(b) is a conceptual representation of Shamir's secret sharing scheme. Both Blakley's and Shamir's scheme are defined over finite fields, respectively $(\mathbb{F}_\ell)^{t+1}$ and $\mathbb{F}_\ell$.

[2]Shamir noted that secret sharing is a more elegant solution to Liu's problem [109] of having 11 scientists work on a secret project of which at least 6 need to be present to handle the documents. For the classical approach with padlocks and keys, it can be shown that the minimal solution requires 462 locks in total and 252 keys per scientist.



(a) Blakley Secret Sharing. Shares are hyperplanes of dimension t, containing the the secret.



(b) Shamir Secret Sharing. Shares are points on a polynomial of degree t with the secret as its free term.

Figure 2.1: Conceptual representation of secret sharing schemes for $t + 1 = 3$.

Both schemes are perfect secret sharing schemes, *i.e.* knowledge of up to t shares does not result in any information about the secret. This can be seen in Fig. 2.1, where for both schemes $t+1 = 3$. For Blakley's scheme, the intersection of two hyperplanes of dimension two results in a line that contains ℓ points, and hence ℓ possible secrets. For Shamir's scheme, one can draw ℓ polynomials of degree two through two points, resulting in $porder$ possible free terms, which define the secret. The dotted lines only represent a few possible polynomials of degree two through two shares.

Later on, two other threshold secret sharing schemes were proposed by Mignotte [116] and Asmuth and Bloom [8], based on the Chinese remainder theorem[3]. However, these schemes are not perfect, since a set of less than $t + 1$ shares contains information about the secret.

The Shamir secret sharing scheme is minimal, *i.e.* the size of the shares is equal to the size of the secret. In contrast, the size of the shares grows with the threshold number in Blakley's scheme. Another advantage of Shamir's scheme over Blakley's[4] is its dynamic ability, that allows us to enhance security, without changing the secret, by handing out new shares of another polynomial through the same point at the origin. This dynamic ability is crucial for proactive security that will be discussed in Chapter 4.

In practice, the polynomial will not explicitly be reconstructed: the secret will be learnt by interpolation of the shares. Any point on a polynomial of degree t

[3]This theorem was first described by Sun Zi in his book *Suanjing – The Mathematical Classic* (third-century AD).

[4]Shares in Blakley's scheme contain the shared secret.

can be reconstructed by constructing the Lagrange form[5] in $t+1$ points of this polynomial. Let $f(z)$ be the generated polynomial, the secret $x = f(0)$ and the shares $x_i = f(i)$. This secret can be reconstructed using a set S of $t+1$ shares as follows:

$$x = \sum_{i \in S} \lambda_i(0) x_i \qquad \text{for} \qquad \lambda_i(k) = \prod_{j \in S, j \neq i} \frac{j-k}{j-i}\,.$$

In the remainder of this thesis the shorter notation λ_i will be used for $\lambda_i(0)$.

2.3.2 Verifiable Secret Sharing

Verifiable secret sharing (VSS) was introduced by Chor *et al.* [40]. A VSS allows the receivers of the shares to verify that the dealer properly shared a secret, i.e. that all shares are evaluations of the same polynomial of degree t. More formally, we briefly rephrase the requirements of a secure VSS (see Pedersen [127] and Gennaro *et al.* [69, Lemma 1]).

Definition 2 (Secure VSS). *A VSS protocol is secure if it satisfies the following conditions:*

1. *Correctness. If the dealer is not disqualified then any subset of $t+1$ honest players can recover the unique secret.*
2. *Verifiability. Incorrect shares can be detected at reconstruction time by using the output of the protocol.*
3. *Secrecy. The view of an adversary $\mathcal{A}$ is independent of the secret.*

Note that $t+1$ honest players are required for correctness. Honest participants do not deviate from the protocol. It might very well be that an honest player is under the influence of the adversary.

The first efficient scheme is due to Feldman [54], building upon Shamir secret sharing. First a polynomial $f(z)$ of degree t is generated and shares $x_i = f(i)$ are handed out to each of the n players over private channels:

$$f(z) = a_0 + a_1 \cdot z + \ldots + a_t \cdot z^t \qquad \text{with} \qquad a_0 = x\,.$$

Second perfect binding, computational hiding commitmentsc_i to the coefficients of this polynomial are broadcasted. To construct these commitments, the

[5]Although named after Joseph Louis Lagrange, it was first discovered in 1779 by Edward Waring and rediscovered in 1783 by Leonhard Euler.

coefficients are hidden in the exponent of the generator g for a group in which the discrete-log assumption holds:

$$c_i = g^{a_i} \qquad \text{for} \quad i = 0 \ldots t-1\,.$$

These commitments allow the shareholders to verify the validity of their share as follows:

$$g^{x_i} = \prod_{j=0}^{t} (c_j)^{i^j}\,.$$

If this verification fails, the shareholder rejects the dealer as faulty.

Pedersen [127] made Feldman's VSS scheme information-theoretically secure, in the sense that the scheme leaks no information on the shared secret, by generating two polynomials

$$f(z) = a_0 + a_1 \cdot z + \ldots + a_t \cdot z^t \qquad \text{with} \qquad a_0 = x$$
$$f'(z) = a'_0 + a'_1 \cdot z + \ldots + a'_t \cdot z^t$$

and handing out shares $x_i = f(i)$ and $x'_i = f'(i)$. The corresponding coefficients are broadcasted as paired commitments, which are known as Pedersen commitments. Let g be a generator for a group in which the discrete-log assumption holds. Let h be another generator of this group for which the discrete logarithm to the base g, and vice versa, is unknown:

$$c_i = g^{a_i} \cdot h^{a'_i} \qquad \text{for} \quad i = 0 \ldots t-1\,.$$

These commitments still allow the shareholders to verify their shares:

$$g^{x_i} \cdot h^{x'_i} = \prod_{j=0}^{t} (c_j)^{i^j}\,,$$

while no information about the shares can be gained from the commitments, as these are perfectly hiding. The binding property is only computational, i.e. it requires the discrete logarithm assumption.

VSS in the asynchronous setting was studied by Cachin *et al.* [32]. Later Zhou *et al.* [175] and Schultz *et al.* [145] also proposed a VSS in this setting.

The above VSS schemes are computational. Unconditional VSS schemes (e.g. proposal Canetti [34]) also exist, however these can only be constructed for $n \geq 3t+1$ [53] as opposed to $n \geq 2t+1$ for computational VSS.

Recently, Backes *et al.* [11] proposed the first computational VSS scheme that can use any cryptographic commitment scheme. All previously proposed VSS schemes rely on the homomorphic properties of the underlying commitment schemes.

2.3.3 Publicly Verifiable Secret Sharing

In a VSS protocol, each receiver can only verify the correctness of his own share. A Publicly Verifiable Secret Sharing (PVSS) protocol allows any (third) party to verify the correctness of all the shares. This also means that complaint procedures or dispute resolution mechanisms are no longer required to disqualify a dishonest dealer.

Stadler [154] was the first to propose a PVSS protocol. In addition to the Feldman commitments, shares are broadcasted in encrypted form and verified using a non-interactive proof of equality of (double) discrete logarithms.

A more efficient protocol was presented by Fujisaki and Okamoto [63], which is secure under a modified RSA assumption.

The first PVSS shown secure under the Decisional Diffie-Hellman assumption was given by Schoenmakers [144]. The shares are broadcasted in encrypted form by hiding them in the exponent of each player's individual public key, which has a different base (another generator) than the Feldman commitments. The dealer then uses non-interactive proofs of discrete-logarithm equality. Furthermore, correct behaviour of the players is verified by extending the secret reconstruction phase with additional proofs of correctness.

Based on Schoenmakers' result Heidarvand and Villar [81] presented the first PVSS protocol where verifiability is obtained from bilinear pairings over elliptic curves; in this approach proofs are no longer needed.

2.3.4 Distributed Key Generation

The drawback of VSS is that a single party knows the secret. This issue can be avoided by generating and sharing the key in a distributed way. The correctness and secrecy requirements for Distributed Key Generation (DKG) protocol were defined by Gennaro et al. [71].

Definition 3 (Secure DKG)**.** *A Distributed Key Generation protocol is secure if it satisfies the following conditions:*
Correctness *is guaranteed if:*
(C1) All subsets of $t+1$ shares provided by honest players define the same private key.
(C2) All honest parties know the same public key corresponding to the unique private key as defined by (C1).
(C3) The private key (and thus also the public key) is uniformly distributed.
Secrecy *is guaranteed if an adversary can learn no information about the private*

key beyond what can be learnt from the public key. This requirement can be further enhanced with a simulation argument: for any adversary there should be a simulator that, given a public key, simulates a run of the protocol for which the output is indistinguishable of the adversary's view of a real run of the protocol that ended with the given public key.

Pedersen [126] used Feldman's VSS to construct the first DKG protocol, sometimes referred to as Joint Feldman, by having each player in a group run an instance of Feldman's protocol in parallel.

Gennaro et al. [69] pointed out that the uniformity of the key produced by Pedersen's DKG protocol cannot be guaranteed in the context of a rushing adversary. A rushing adversary can wait in each communication round to send messages on behalf of the corrupted devices until he has received the messages from all uncorrupted devices. They constructed a new DKG protocol by first running Pedersen's VSS in parallel (Joint Pedersen). Since Pedersen VSS does not produce a public key, an extra round of communication, namely an instance of Joint Feldman on the first polynomial, has to be added to compute the public key. They proved their protocol secure against a static adversary by means of a simulation argument. Interestingly, Gennaro et al. showed later [70] that, despite the biased distribution of the key, certain discrete-log schemes that use Pedersen DKG can still be proved secure at the cost of an increased security parameter.

Canetti et al. [36] used interactive knowledge proofs and erasures, i.e., players erase private data before commitments or public values are broadcasted, in the key construction phase of the DKG of [69] to make the protocol secure against adaptive adversaries. Comparable adaptively secure threshold schemes were presented by Frankel et al. [59].

In the protocols discussed so far, it is assumed that there are private channels between each pair of players. Both [36] and [59] suggest that these channels can still be established even with an adaptive adversary using the non-committing encryption technique of Beaver and Haber [15], which assumes erasures. Jarecki and Lysyanskaya [92] criticised this erasure model and pointed out that the protocols presented in [36] and [59] are not secure in the concurrent setting, i.e., two instances of the same scheme cannot be run at the same time. They solved this by introducing a “committed proof”, i.e., a zero-knowledge proof where the statement that is being proved is not revealed until the end of the proof. To implement the secure channels without erasures they use an encryption scheme that is non-committing to the receiver. Abe and Fehr [3] later proposed an adaptively-secure (Feldman-based) DKG and applications with complete security proofs in the Universal Composability framework of Canetti [35]. They demonstrated that a discrete-log DKG protocol can be achieved without interactive zero-knowledge proofs. However, they still need a single inconsistent player and a secure message transmission functionality (private channels), which can be realised using a receiver non-committing transmission protocol based on [92].

The first DKG that does not require private channels was given by Fouque and Stern [56]. The main buildings blocks for their construction are the Paillier cryptosystem and a new non-interactive zero-knowledge proof. To deal with a rushing adversary it is simply assumed that communication is completely synchronous. For participants not present during the DKG the amount of information that needs to be stored, i.e., the subshares that need to be decrypted, is linear in the number of participants that are active in the DKG.

Kate and Goldberg [100] were the first to study DKG in an asynchronous setting and proposed a DKG that is deployable over the internet.

2.4 Threshold Cryptography on Personal Devices

The mobility of personal devices is usually considered as a major benefit. However, this mobility is a weakness in terms of security and reliability. Mobile devices are susceptible to theft, they can easily be forgotten or lost, or simply run out of battery power. These weaknesses can be mitigated by introducing threshold cryptography on personal devices.

For access control scenarios, access will only be granted when at least the threshold number of personal devices is present. One can also encrypt the sensitive data on one's personal devices, and only allow decryption when the threshold number of personal devices is present. The first case will offer some protection (the adversary needs to compromise at least the threshold number of devices) against the theft of credentials, while the second case offers protection against data theft. This is the security property.

For access control scenarios, the end-user can still gain access as long as he carries any subset of his personal devices, of size at least the threshold number. Similarly he will still be able to decrypt his sensitive data on his personal devices. We define resilience as the number of absent devices that the system can tolerate while remaining functional.

2.4.1 Communication and Adversarial Model

We assume that devices communicate over a dedicated broadcast channel[6]. By dedicated we mean that if a device broadcasts a message, then it is received by all other devices within communication range. For communication between personal devices, which form a small local network, this assumption is not unreasonable. There are no private channels: all communication goes over the broadcast channel. Communication is round-synchronous, protocols run in rounds and there is a time bound on each round.

In certain cases, the user forms an Out-Of-Band (OOB) channel between the devices. This channel has the benefit of being authentic, but the downside is that this channel has only limited bandwidth.

A distinction is commonly made between static and adaptive adversaries. Static means that the adversary corrupts the devices before the protocol starts, whereas adaptive means that a device can become corrupt before or at any time during execution of the protocol. We assume a malicious computationally bounded static adversary that can corrupt up to t out of n devices. The adversary has access to all information stored by the corrupted devices and can manipulate their behaviour during the execution of a protocol in any way. The round-synchronous communication implies that the adversary could be rushing.

2.4.2 Threshold

When considering how to set the threshold, we need to take into account the adversary that compromises τ devices. As long as the adversary corrupts fewer than the threshold number of devices, the adversary will not learn any information on the shared private key (apart from the information that is contained in the corresponding public key). This results in the condition on overall security:

$$\tau < t + 1\,. \tag{2.1}$$

The overall system needs to remain functional in presence of adversaries. To prevent a permanent Denial of Service (DoS) attack, enough uncompromised devices should remain active to allow reconstruction the private key (or use the private key implicitly, e.g., to decrypt a ciphertext or place a signature). This results in the condition on overall resilience:

$$n - \tau \geq t + 1 \qquad \text{or} \qquad \tau < n - t\,. \tag{2.2}$$

[6]We abstract from the actual implementation of the dedicated broadcast channel.

Our adversarial model assumes that the adversary can compromise up to t out of n devices. To prevent a worst case adversary, that compromises $\tau_{max} = t$ devices, from mounting a permanent DoS attack, the following boundary on the threshold is obtained:

$$n \geq 2t + 1 \qquad \text{or} \qquad t + 1 \leq \frac{n+1}{2}\,. \tag{2.3}$$

This corresponds with the intuitive feeling that, given an honest majority, overall security and resilience can be achieved. From a security point of view the threshold should be as close to this boundary as possible. Meaning that for an even number of devices, the threshold consists of exactly half the devices. For an odd number of devices, the threshold represents the smallest majority.

2.5 Conclusion

This chapter gave an overview of the general constructs related to threshold cryptography, including how to set up a secret sharing scheme and how to subsequently use the shared secret for cooperative decryption or generation of signatures. We introduced the benefits of threshold cryptography for the protection of personal devices. In this context, we introduced a general communication and adversarial model and discussed the optimal choice for the threshold number. The communication and adversarial model will be used in the subsequent chapters.

Chapter 3

Threshold Cryptography on Less Powerful Devices

Many personal devices are not suitable to be included in threshold schemes, because these do not offer (secure) storage, which is needed to store shares of the private key. This chapter presents a novel way of storing shares and shows how to transform several cryptographic protocols accordingly. Following this approach, threshold schemes can also include low-cost devices with a factory-embedded key, e.g., car keys and access cards.

3.1 Introduction

In the previous chapter we introduced the concept of threshold cryptography and its benefits when deployed on personal devices. The benefits are increased security, because an adversary needs to compromise multiple personal devices; and resilience, since for an end-user a subset of his personal devices is sufficient. This chapter focusses on resilience. The system can tolerate the absence of up to $n - t - 1$ personal devices[1]. By increasing the number of parties in the threshold scheme, the system's overall resilience increases. This means that more personal devices can be forgotten, broken, stolen, ..., while the end-user still enjoys a functional security system.

However, the number of personal devices suitable for threshold schemes is limited because many of these do not incorporate (secure) storage, which is

[1]Recall that n is the total number of personal devices and $t + 1$ the threshold number.

needed to store shares of the private key. We enlarge the group of high-end devices, e.g., tablets or smart phones, by also considering small devices with public-key functionality, e.g., car keys or access cards. Typically, these small devices have a factory-embedded private key, which cannot be updated and is the only object that resides in tamper-proof secure storage[2]. Furthermore, these devices only have limited computational power.

We propose to store shares in protected form. In this form, these can be stored can be as well on the device as externally. This is a desirable property for devices that do not even have writeable memory, as is the case for most car keys. These protected shares are generated through a run of our new Distributed Key Generation (DKG) protocol, which is publicly verifiable. Public verifiability implies that the correctness of any device's contribution can be verified by all participants and any third party. As such, not every device needs to be present during the DKG. Moreover, shares can be used implicitly, as these are never needed in unprotected form. Furthermore, some devices can be completely ignorant of the underlying threshold scheme and only serve as partial decryption oracles. We demonstrate how protected shares can easily be used in discrete-log based cryptosystems and signature schemes. More specifically, we demonstrate this for the ElGamal [66] and the Cramer-Shoup [43] cryptosystems, and the Schnorr signature scheme [142].

3.2 Protecting Shares

For the group's private key $x \in \mathbb{Z}_\ell$ to be shared, each device receives a share $x_i \in \mathbb{Z}_\ell$ that has to be (securely) stored. Since not all devices provide (secure) storage, there is a need for an alternative approach. We propose to store these shares in protected form.

Before discussing how to construct the protected shares (Setup), we first give a formalisation of the public parameters and describe how the devices generate their public/private key pairs (Pre-Setup).

3.2.1 Pre-Setup

- **Init(1^k):** The input is a security parameter k. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be finite cyclic groups of prime order ℓ with P, Q and $g = \hat{e}(P, Q)$ generators of the respective groups. It is assumed that there is no known efficiently

[2]Private keys can be securely stored in hardware by making use of *Physically Unclonable Functions* (PUF) [77]. Small differences during the fabrication process allows us to derive a unique private key with only a slightly higher fabrication cost.

computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$. Let P' be another generator of $\mathbb{G}_1$ for which the discrete logarithm relative to the base P, and vice versa, is unknown and let $g_1 = g$, $g_2 = \hat{e}(P', Q)$.

The initialisation procedure outputs the description of the groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and the pairing $(\hat{e})$ along with the public system parameters

$$PubPar = (P, P', Q) \in \mathbb{G}_1^2 \times \mathbb{G}_2 \,.$$

For some protocols, an additional generator $P'' \in \mathbb{G}_1$ is required for which the discrete logarithms relative to the base P and P', and vice versa, are unknown. In this case we also define $g_3 = \hat{e}(P'', Q)$.

- **KeyGen(*PubPar*,$\mathcal{D}_i$):** For the given device $\mathcal{D}_i$ a random $s_i \in_R \mathbb{Z}_\ell^*$ is chosen as private key. The corresponding public key is $S_i = s_i Q$. The key generation procedure outputs $\mathcal{D}_i$'s key pair

$$(s_i, S_i) \in \mathbb{Z}_\ell^* \times \mathbb{G}_2 \,.$$

Note that the key generation procedure is executed only once in the lifetime of each device $\mathcal{D}_i$ and that s_i is the only secret that has to be securely stored. Typically, this routine is executed during production or personalisation of the device. Another public key can easily be computed for a different set of system parameters if this is required.

3.2.2 Setup

These key pairs (s_i, S_i) will be used to protect the shares for the devices. An obvious answer would be to store a share encrypted under the device's public key. At some point the share will need to be decrypted and appear in the clear in unprotected memory, which is undesirable. Furthermore this would involve a costly decryption operation every time a device wants to use its share. Another option is to store the share as the product $x_i s_i$. The obvious disadvantage is that $t + 1$ devices can collaborate to compute another device's private key s_i.

As we do not want a device's private key s_i ever to be revealed, we will combine shares with the device's public key $C_i = x_i S_i \in \mathbb{G}_2$. These will be denoted as public correction factors. A similar idea was used by Schoenmakers [144]. However, here we use bilinear pairings to achieve public verifiability and easy integration of our scheme in existing discrete-log cryptosystems and signature schemes, without ever having to reveal the shares (see Sect. 3.4). We define[3] the

[3]Note that we could use the notation X and X_i for the private key and its shares. However, to compute the correction factor C_i, elements of $\mathbb{Z}_\ell$ will be combined with S_i, but by definition, private key material is in $\mathbb{G}_2$.

group's private key as $xQ \in \mathbb{G}_2$ and its public key as $y = g^x = \hat{e}(P, xQ) \in \mathbb{G}_T$. Hence, the share of a device is $x_i Q = s_i^{-1} C_i \in \mathbb{G}_2$. The construct group routine is formally defined as follows:

- **ConstructGroup(*PubPar*,$\{\mathcal{D}_i, S_i\}$,*t*):** A subset of the devices $\mathcal{D}_i$ generates the group's public key g^x and shares the private key xQ in the form of public correction factors $C_i = x_i S_i$ for all n devices. The procedure outputs the group's public key

  $$y = g^x = \hat{e}(P, xQ) \in \mathbb{G}_T$$

  and the public correction factors which are added to the public parameters

  $$PubPar = (P, P', Q, y, \{C_i\}_{i=1,\ldots,n}) \in \mathbb{G}_1^2 \times \mathbb{G}_2 \times \mathbb{G}_T \times \mathbb{G}_2^n \,.$$

ConstructGroup can be implemented either by a trusted dealer or in a distributed manner. When choosing for a trusted dealer, implementation of this routine is straight-forward. To construct the public correction factors, the trusted dealer runs an instance of Shamir's secret sharing with appropriate parameters to obtain the shares x_i and then multiply these with the corresponding public keys S_i. In the next section, we present how the ConstructGroup can be implemented in a distributed manner by using our proposed DKG protocol.

3.3 Distributed Key Generation

In this section, we present a new distributed key generation (DKG) protocol. Recall that we want to set up a threshold construction without the devices having to (securely) store their share.

Since a DKG usually requires intensive computations from all devices involved, it is desirable that the less powerful devices do not need to contribute. This can be achieved by only using public channels and introducing public verifiability. An added bonus of public verifiability is that our DKG does not need a complaint procedure or dispute resolution mechanism.

Our DKG consists of two phases. First, a private key is jointly generated and shared through the parallel execution of a new publicly verifiable secret sharing (PVSS) protocol. This PVSS protocol is described in Sect. 3.3.1. Second, the corresponding public key is extracted. Together, these two phases make up our new DKG protocol, which is presented in Sect. 3.3.2.

3.3.1 Publicly Verifiable Secret Sharing

The main building block to construct our DKG protocol is a new PVSS protocol. In this protocol, a dealer generates shares of a secret and distributes them in protected form. Any party observing the output of the protocol can verify that the dealer behaved correctly. Our proposed PVSS protocol can be described as follows.

The dealer chooses uniformly at random $x \in_R \mathbb{Z}_\ell$. The actual secret that will be shared at the end of the protocol is xQ. Similar to Pedersen's VSS scheme [127], the dealer chooses two random polynomials f and f' of degree t, sets the constant term of the first polynomial to x and broadcasts pairwise commitments $A_k \in \mathbb{G}_1$ to the coefficients of the polynomials. Evaluations of both polynomials will be combined with the public keys of the devices and broadcasted in protected form. Each device then verifies that all broadcasted shares are correct by applying the pairing to check them against the commitments. The details of the protocol are given in Fig. 3.1.

Private channels are avoided because the shares x_iQ are broadcasted in protected form x_iS_i. Each device could recover its share by using its private key. However, the shares are never needed in unprotected form. The protected form allows for public verifiability, since for any device $\mathcal{D}_i$ the correctness of x_iS_i and x'_iS_i can be verified by pairing the commitments A_k with $\mathcal{D}_i$'s public key S_i. The dealer is disqualified, if for any $\mathcal{D}_i$ this verification fails. As a consequence, there is no need for a cumbersome complaint procedure. Moreover, not all devices need to be present during the execution of the protocol to get and verify their received shares. The broadcasted protected shares can be stored as such by the present devices.

In the next theorem we will demonstrate that our new PVSS protocol satisfies the requirements of secure VSS protocol as given by Def. 2.

Theorem 2. *Our new PVSS protocol is a secure VSS protocol (Def. 2) under the divisional variant of the DDH assumption in $\mathbb{G}_2$.*

Correctness. It follows directly from Pedersen's result [127] that each subset of $t+1$ devices can reconstruct the coefficients c_kQ, c'_kQ of the polynomials $F(z) = f(z)Q$ and $F'(z) = f'(z)Q$ from their shares. If the dealer is not disqualified then Eqn. (3.1) holds for all devices and the coefficients will successfully be verified against the commitments A_k. Hence, it can be verified that all shares are on the same (respective) polynomial and each subset of $t+1$ devices can compute the same secret $xQ = F(0)$. $\square$

Input: *PubPar*, the set of participating devices $\mathcal{D}_i$ and public keys S_i, $t+1$
Output: Protected shares $x_i S_i$, such that $x_i Q$ is the share of device $\mathcal{D}_i$

The dealer shares the secret xQ, for which he chooses $x \in_R \mathbb{Z}_\ell$:

1. The dealer constructs two polynomials $f(z)$ and $f'(z)$ of degree t by choosing random coefficients $c_k, c'_k \in_R \mathbb{Z}^*_\ell$ for $k = 0 \dots t$, except for c_0, which is $c_0 = x$:
$$f(z) = c_0 + c_1 z + \cdots + c_t z^t \quad , \quad f'(z) = c'_0 + c'_1 z + \cdots + c'_t z^t \,.$$
The dealer broadcasts commitments
$$A_k = c_k P + c'_k P' \quad , \quad k = 0 \dots t \,.$$
2. For each device $\mathcal{D}_i$, the dealer computes and broadcasts
$$x_i S_i \ , \ x'_i S_i \quad \text{with} \quad x_i = f(i) \ , \ x'_i = f'(i) \quad , \quad i = 1 \dots n \,.$$
3. Each device verifies the broadcasted shares for all $\mathcal{D}_i$ by checking that
$$\hat{e}(P, x_i S_i) \cdot \hat{e}(P', x'_i S_i) = \prod_{k=0}^{t} \hat{e}(A_k, S_i)^{i^k} \,. \tag{3.1}$$
If any of these checks fails, the dealer is disqualified.

Figure 3.1: Publicly Verifiable Secret Sharing with Protected Shares.

Verifiability. During reconstruction $\mathcal{D}_i$ provides $x_i Q$ and $x'_i Q$, and it can be verified that
$$\hat{e}(P, x_i Q) \cdot \hat{e}(P', x'_i Q) = \prod_{k=0}^{t} \hat{e}(A_k, Q)^{i^k} \,.$$
□

Secrecy. Consider a worst-case static adversary $\mathcal{A}$, i.e., an adversary that corrupts t devices before the protocol starts. The protocol is semantically secure against $\mathcal{A}$, if $\mathcal{A}$ chooses two values $x_0 Q, x_1 Q \in \mathbb{G}_2$ and cannot determine which of these two was shared with negligible advantage over random guessing, given the output of a run of the protocol that shared either the secret $x_0 Q$ or $x_1 Q$. We prove the semantic security by showing that no such adversary can exist.

If there exists an $\mathcal{A}$ that has a non-negligible advantage in attacking the semantic security of our protocol, then we can build a simulator SIM that uses $\mathcal{A}$ to solve an instance of the divisional DDH problem in $\mathbb{G}_2$ (see Sect. 1.5.3). Since this is assumed to be a hard problem we conclude that no such adversary can exist.

We now describe this simulator. The simulator SIM is given a tuple $\langle Q, aQ, cQ, abQ\rangle$ and has to decide if this is a valid DH tuple, i.e., if $cQ = bQ$.

1. The simulator SIM does the pre-setup. He chooses the system parameters PubPar, which contain P and $P' = \eta P$, with η known to SIM. He constructs a set of devices $\mathcal{D}_i$, of which one will be the designated device, denoted as $\mathcal{D}_d$. For each $\mathcal{D}_i \neq \mathcal{D}_d$, SIM generates a random key pair. The public key of $\mathcal{D}_d$ is set to $S_d = cQ$.

2. The adversary $\mathcal{A}$ receives PubPar and the set of devices along with their public keys. He announces the subset of corrupted devices, which will be denoted by $\mathcal{D}_j$ for $j = 1 \ldots t$.

3. The simulator SIM gives the private keys s_j of the corrupted devices to $\mathcal{A}$. Device $\mathcal{D}_d$ is corrupted with a worst-case probability of roughly 1/2, in which case the simulation fails.

4. $\mathcal{A}$ outputs two values x_0Q and x_1Q, of which one has to be shared.

5. Without loss of generality, we assume SIM chooses x_0Q. The output of the VSS protocol is generated as follows.

   - SIM chooses k random values $z_k \in_R \mathbb{Z}_\ell^*$ and broadcasts commitments $A_k = z_kP$.
   - SIM constructs a random polynomial $F(z)$ of degree t subject to $F(0) = x_0Q$ and $F(d) = aQ$. For Eqn. (3.1) to hold, future shares x_iQ and x'_iQ will have to satisfy

   $$\alpha_i Q = x_i Q + \eta x'_i Q \quad \text{with} \quad \alpha_i = \sum_{k=0}^{t} z_k i^k\,. \tag{3.2}$$

     SIM evaluates the polynomial $F(z)$ and sets the shares $x_jQ = F(j)$ for each corrupted $\mathcal{D}_j$. For the non-corrupted $\mathcal{D}_i \neq \mathcal{D}_d$, SIM chooses random shares $x_iQ \in_R \mathbb{G}_2$. For $i \neq j$, the shares on the second polynomial x'_iQ and x'_jQ are determined by Eqn. (3.2).
   - With the private keys s_i and s_j, SIM computes the protected shares x_iS_i, x'_iS_i and x_jS_j, x'_jS_j.
   - For the designated device, SIM sets $x_dS_d = abQ$ and $x'_dS_d = \eta^{-1}(\alpha_dS_d - abQ)$.

- All protected shares are broadcasted by SIM.

6. The adversary outputs a guess to which of the secrets was shared. If $\mathcal{A}$ has a non-negligible advantage in determining which secret was shared then SIM concludes that $\langle Q, aQ, cQ, abQ \rangle$ must be a valid DH tuple.

The view of $\mathcal{A}$ consists of the commitments A_k, all public keys, the private keys of the corrupted devices, all protected shares and the shares of the corrupted devices. The adversary $\mathcal{A}$ can only gain an advantage in guessing which key was shared from values, other than his own shares, which were not chosen at random. This leaves him with only his shares x_jQ and the values x_dS_d and S_d. The adversary's problem of deciding which secret was shared is equivalent to deciding whether $x_dQ = x_0Q - \sum \lambda_j x_j Q$ or $x_dQ = x_1Q - \sum \lambda_j x_j Q$. Because we assume SIM chose x_0Q, $\mathcal{A}$ has to decide whether $\langle Q, x_0Q - \sum \lambda_j x_j Q, S_d, x_dS_d \rangle$ is a valid DH tuple or not.

$\square$

We note that given the specific form in which the shares are broadcasted, our PVSS protocol cannot be proved secure against an adaptive adversary by means of a simulation argument, which does not imply that it is insecure. Indeed, it was already suggested by Canetti *et al.* [36] and Frankel *et al.* [59] that to maintain private transmission of shares some form of non-committing encryption should be used. We insist on storing shares as x_iS_i in order to maintain the nice properties of this form, which will allow us to integrate our construction in other threshold applications, as shown in Sect. 3.4.

A somewhat related[4] PVSS scheme was presented by Heidarvand and Villar [81]. Our PVSS scheme differs from theirs by putting the secret in $\mathbb{G}_2$, instead of $\mathbb{G}_T$, and thus allowing it to be a building block for DKG and discrete-log constructions. Moreover, our protocol is semantically secure while the scheme in [81] is only proved to be secure under a weaker security definition, because the adversary is not allowed to choose the secrets that it has to distinguish.

[4]Note that we use an asymmetric pairing which is more standard (e.g., see [65]) than the symmetric form used in [81].

3.3.2 Distributed Key Generation

We now establish a new DKG protocol that outputs protected shares and is publicly verifiable. Inspired by Gennaro *et al.* [69] and Canetti *et al.* [36] the protocol consists of two phases. In the first phase, the group's private key is generated distributively and shared through a joint PVSS. In the second phase, the group's public key is computed. This phase follows to a large extent the result of Canetti et al. [36]. The protocols proceeds as follows.

Each participating device runs an instance of our new PVSS protocol. It chooses a secret $c_{i,0} \in_R \mathbb{Z}_\ell$ and broadcasts shares of that secret in protected form. These will be denoted as protected subshares. Each device, acting as a dealer, that is not disqualified is added to a set of qualified devices, denoted as QUAL. Each participating device constructs this set using public data that is available to all participating devices, which results in identical sets across the participating devices. As long as the cardinality of QUAL is smaller than $t+1$, the qualified devices wait for more qualified devices or abort the protocol eventually. The group's private key, although never computed explicitly, is defined as $xQ = \sum_{i \in \mathrm{QUAL}} c_{i,0}Q$. A device's protected share $C_i = x_i S_i$ is computed as the sum of the protected subshares that were received from the devices in QUAL.

To recover the group's public key $y = g^x$, the qualified devices will expose $g^{x_i} = \hat{e}(s_i^{-1}P, C_i)$ from which y can easily be computed through Lagrange interpolation. Opposed to [36], we do not expose $g^{c_{i,0}}$, which avoids the costly reconstruction of the $g^{c_{j,0}}$ of the qualified devices that no longer participate in the second phase. Each device will prove in zero-knowledge that the exponent of g^{x_i} matches the share x_iQ hidden in C_i, without revealing it. The technique used in our protocol is a committed proof as proposed by Cramer *et al.* [42]. An earlier version of our work was based on interactive zero-knowledge proofs in which the uniformly distributed challenges were generated by another run of our Joint PVSS. This alternative key extraction will be discussed later on.

Cramer *et al.* [42] proposed a very efficient method of extending Σ-protocols, which are used for zero-knowledge proofs. They make use of a committed proof, a zero-knowledge protocol where the statement that is being proved is not revealed until the end of the proof. To create the (almost) random common challenge with multiple parties, each party contributes a random string. The concatenation of these strings will contain at least one input from parties that are not under the influence of the adverary and therefore contain randomness. At the time of creating the challenge, all parties already committed to the statement they want to prove, without other parties having knowledge about the statement.

The details of the protocol are given in Fig. 3.2. Note that, since no explicit private channels are required, only a minimum of $t+1$ honest devices must participate in the DKG.

Our DKG protocol withstands the attack of a rushing adversary that can influence the distribution of the group's key as described by Gennaro et al. [71]. In this attack an adversary is able to compute a deterministic function of the private key from the broadcasts, before sending out his contributions. He can influence the set of qualified devices by choosing whether or not to send out proper contributions. This allows influencing the outcome of the deterministic function by at most t bits and thus the distribution of the private key. Since no mapping from the search space for the public key to the search space for the private key is known, this attack is more of theoretical nature. However, the non-uniformity of the private key is a problem for most signature schemes and cryptosystems, since their security depends partly on the uniform sampling of a private key. In our protocol, no such function can be computed before the second phase. But, because the private key and thus also the correction factors are fixed after the first phase and determined by QUAL, the adversary can no longer influence the group's key. As long as $t+1$ honest devices participate, the public key can be recovered in the second phase.

We now prove that our new DKG protocol is a secure DKG protocol according to the requirements specified in Def. 3.

Theorem 3. *Our new DKG protocol is a secure DKG protocol (Def. 3) under the divisional variant of the coDBDH-2 assumption.*

Correctness. All honest devices construct the same set of qualified devices QUAL since this is determined by public broadcasted information.

- (C1) Each $\mathcal{D}_i$ that is in QUAL at the end of phase 1 has successfully shared $c_{i,0}Q$ through a run of our PVSS protocol. Any set of $t+1$ honest devices $\mathcal{D}_i$ that combine correct shares x_jQ can reconstruct the same secret xQ since

$$\begin{aligned} xQ &= \sum_{i\in\text{QUAL}} c_{i,0}Q = \sum_{i\in\text{QUAL}}\left(\sum_j \lambda_j x_{ij}Q\right) = \sum_j \lambda_j \sum_{i\in\text{QUAL}} x_{ij}Q \\ &= \sum_j \lambda_j x_j Q\,. \end{aligned}$$

  In the key extraction phase of our protocol at least $t+1$ values g^{x_i} have been exposed and thus, by interpolation, g^{x_j} can be computed for any $\mathcal{D}_j$. This allows us to tell apart correct shares from incorrect ones.

Input: *PubPar*, the set of participating devices $\mathcal{D}_i$ and public keys S_i, $t+1$
Output: Protected shares $C_i = x_i S_i$, the group's public key $y = g^x$

1. Distributed generation of protected shares. See Fig. 3.3.
2. Public key extraction. See Fig. 3.4.

Figure 3.2: DKG with Protected Shares.

All participating devices $\mathcal{D}_i$ run the PVSS protocol simultaneously, the protected subshares are only broadcasted after receiving all commitments from all participating devices.

(a) Each $\mathcal{D}_i$ constructs two polynomials $f_i(z)$ and $f'_i(z)$ of degree t by choosing random coefficients $c_{i,k}, c'_{i,k} \in_R \mathbb{Z}^*_\ell$ for $k = 0 \dots t$:

$$f_i(z) = c_{i,0} + c_{i,1}z + \cdots + c_{i,t}z^t \quad , \quad f'_i(z) = c'_{i,0} + c'_{i,1}z + \cdots + c'_{i,t}z^t \,,$$

and broadcasts commitments

$$A_{i,k} = c_{i,k}P + c'_{i,k}P' \quad \text{for} \quad k = 0 \dots t \,.$$

(b) For each device $\mathcal{D}_j$, each $\mathcal{D}_i$ computes and broadcasts

$$x_{ij}S_j \ , \ x'_{ij}S_j \quad \text{with} \quad x_{ij} = f_i(j) \ , \ x'_{ij} = f'_i(j) \,.$$

(c) Each device verifies the broadcasted shares for all $\mathcal{D}_i$ by checking that

$$\hat{e}(P, x_{ij}S_j) \cdot \hat{e}(P', x'_{ij}S_j) = \prod_{k=0}^{t} \hat{e}(A_{i,k}, S_j)^{j^k} \,.$$

Each $\mathcal{D}_i$ that is not disqualified as a dealer is added to the list of qualified devices, denoted by QUAL. The protocol halts until$|\text{QUAL}| \geq t+1$. The group's private key is defined as $xQ = \sum_{i \in \text{QUAL}} c_{i,0}Q$. For each $\mathcal{D}_i$ its protected share is computed as

$$C_i = x_i S_i = \sum_{j \in \text{QUAL}} x_{ji} S_i \,.$$

Figure 3.3: Distributed Generation of Protected Shares.

The qualified devices expose g^{x_i} to compute the public key $y = g^x$.

(a) Each $\mathcal{D}_i$ in QUAL computes $\alpha_i = g^{x_i} = \hat{e}(s_i^{-1}P, C_i)$ and $A_i = s_i P''$. In addition, $\mathcal{D}_i$ chooses a random $r_i \in_R \mathbb{Z}_\ell^*$ and computes $\beta_i = g^{r_i}$ and $B_i = r_i S_i$.

- Commit to the values $\alpha_i, \beta_i, A_i, B_i$. First these values are converted to binary strings and concatenated. A hash function is used to get an element of Z_l^*. For a random $d_i' \in_R \mathbb{Z}_\ell^*$:

$$D_i = d_i P + d_i' P' \quad \text{with} \quad d_i = H(\alpha_i \parallel \beta_i \parallel A_i \parallel B_i)\,.$$

- Provide randomness for the challenge $e_i \in_R \mathbb{Z}_\ell^*$.

$\mathcal{D}_i$ in QUAL broadcasts the values D_i and e_i.

(b) Generation of the challenge e. All broadcasted e_i are concatenated and put through a hash function to get an element of Z_l^*.

(c) All $\mathcal{D}_i$ in QUAL open their commitments by broadcasting $\alpha_i, \beta_i, A_i, B_i, d_i'$. Additionally they broadcast $Z_i = s_i^{-1}(r_i S_i + eC_i) = (r_i + ex_i)Q$ which completes the zero knowledge protocol.

(d) Any device can verify that

$$D_i = d_i P + d_i' P' \quad \text{with} \quad d_i = H(\alpha_i \parallel \beta_i \parallel A_i \parallel B_i) \qquad \text{and}$$

$$\hat{e}(A_i, Q) = \hat{e}(P'', S_i), \quad \hat{e}(P, Z_i) = \alpha_i^e \beta_i, \quad \hat{e}(A_i, Z_i) = \hat{e}(P'', B_i + eC_i)\,.$$

(e) The public key y is computed from $t+1$ correctly verified $\alpha_i = g^{x_i}$ as

$$y = \prod \alpha_i^{\lambda_i}\,.$$

Figure 3.4: Public Key Extraction.

- (C2) This property follows immediately from the key extraction phase and the relation between the $c_{i,0}Q$ and the shares x_iQ given for the previous property (C1).
- (C3) The private key is defined as $xQ = \sum_{i \in QUAL} c_{i,0}Q$ and each $c_{i,0}Q$ was shared through an instance of our PVSS. Since we proved that a static adversary cannot learn any information about the shared secret, the private key is uniformly distributed as long as one non-corrupted device successfully contributed to the sum that defines xQ.

□

Secrecy. We describe a simulator SIM that, given a public key y, simulates a run of the protocol and produces an output that is indistinguishable from the adversary's view of a real run of the protocol that ended with the given public key. We assume that SIM knows $\eta \in \mathbb{Z}_\ell^*$ for which $P' = \eta P$.

- The first phase of the DKG is run as in the real protocol. Since SIM knows the private keys s_i of at least $t+1$ non-corrupted devices, he knows at least $t+1$ shares $x_iQ = s_i^{-1}C_i$. By interpolation of these shares, SIM learns the shares of the corrupted devices. This also allows SIM by pairing to compute g^{x_i} for all devices.
- In the second phase of the DKG protocol SIM sets $g^{x_i^*}$ for the non-corrupted $\mathcal{D}_i$, such that the public key will be y. The $g^{x_i^*}$ for the non-corrupted $\mathcal{D}_i$ are calculated by interpolation of the g^{x_j} of the corrupted $\mathcal{D}_j$ and $y = g^x$.

  (a) SIM broadcasts $D_i = d_i'P$ and e_i for each non-corrupted $\mathcal{D}_i$.

  (b) The challenge for the zero knowledge proof e is constructed.

  (c) For all non-corrupted $\mathcal{D}_i$, SIM sets $\alpha_i^* = g^{x_i^*}$, $A_i = s_iP''$ and computes $\beta_i^* = \alpha_i^{-e}g^{z_i}$, $B_i^* = z_iS_i - eC_i$ and $Z_i^* = z_iQ$ for a random $z_i \in_R \mathbb{Z}_\ell^*$. The new value $d_i'^*$ is computed as $\eta^{-1}(d_i' - d_i^*)$ with $d_i^* = H(\alpha_i^* \parallel \beta_i^* \parallel A_i \parallel B_i^*)$ such that $D_i = d_i^*P + d_i'^*P'$. SIM broadcasts $(\alpha_i^*, \beta_i^*, A_i, B_i^*, d_i^*, Z_i^*)$ for each non-corrupted $\mathcal{D}_i$.

  (d) All values verify.

$$\hat{e}(P, Z_i^*) = g^{z_i} \quad \text{and} \quad \alpha_i^{*e}\beta_i^* = \alpha_i^{*e}\alpha_i^{*-e}g^{z_i} = g^{z_i},$$

$$\hat{e}(A_i, Z_i) = g^{s_iz_i} \quad \text{and} \quad \hat{e}(P'', B_i^* + eC_i) = \hat{e}(P'', z_iS_i) = g^{s_iz_i}.$$

  (e) At the end of the protocol the public key is computed as the given y.

To prevent an adversary from being able to distinguish between a real run of the protocol and a simulation, the output distribution must be identical. The first phase, i.e., the Joint PVSS, is identical in both cases. The data that are output in the second phase and that have a potentially different distribution in a real run and simulation are given in the following table. We show that all data in this table have a uniform distribution.

	REAL	SIM
1.	g^{x_i}	$g^{x_i^*}$
2.	$g^{r_i}, r_i S_i$	β_i^*, B_i^*
3.	d_i'	$d_i'^*$
4.	Z_i	Z_i^*

1. The values x_i are evaluations of a polynomial of degree t with uniformly random coefficients. The values x_i^* are evaluations of a polynomial that goes through t evaluations of the first polynomial, namely the x_j of the corrupted participants, and through the discrete logarithm of y. Since the protocol is assumed to generate a uniformly random key, the new polynomial's distribution is indistinguishable from the distribution of the first.

2. The value r_i was chosen uniformly at random. In the simulation $\beta_i^* = g^{z_i}(g^{x_i^*})^{-e}$ and $B_i^* = z_i S_i - eC_i$. The value z_i is uniformly random.

3. In the simulation $d_i'^* = (d_i' - d_i^*)\eta^{-1}$, the value d_i' was chosen uniformly at random.

4. We have that $Z_i = r_i Q + \eta x_i Q$ and $Z_i^* = z_i Q$. The values r_i, d and z_i were chosen uniformly at random.

Even if the modified $g^{x_i^*}$ have the right output distribution, it is important to note that by broadcasting the modified $g^{x_i^*}$ we introduce a new assumption. Namely that an adversary cannot distinguish between $\langle P, Q, x_i s_i Q, s_i Q, g^{x_i}\rangle$ and $\langle P, Q, x_i s_i Q, s_i Q, g^{x_i^*}\rangle$. This is the divisional variant of the coDBDH-2 assumption, as defined in Sect. 1.5.3.

$\square$

Alternative Public Key Extraction

An alternative version of our DKG protocol uses interactive zero-knowledge proofs to prove the validity of the values g^{x_i} exposed by the devices in the key extraction phase. These proofs require uniformly distributed and unpredictable challenges. Please note that this challenge can be the same for all proofs. The devices distributively generate a proper challenge as follows.

A uniformly distributed challenge is generated through another run of our joint PVSS. By distributed generation of the challenge, at least one honest device contributes. This guarantees that the challenge will be unpredictable and uniformly distributed. All devices receive protected shares d_iS_i and d'_iS_i. After open reconstruction, for which each device uses its private key to broadcast the unprotected shares d_iQ and d'_iQ, we have a uniformly distributed element $dQ \in \mathbb{G}_2$. However, the challenge needs to be some element $\tilde{d} \in \mathbb{Z}_\ell$. This implies a bijective (not necessarily homomorphic) mapping $\psi : \mathbb{G}_2 \rightarrow \mathbb{Z}_\ell$. An example of such a mapping is to take the x-coordinate of dQ modulo ℓ, as is used in ECDSA signatures. Several issues have been reported with this mapping and alternatives, e.g., taking the sum of the x and the y-coordinates modulo ℓ [112], have been proposed. We refer the reader to [29] for an in-depth treatment of this subject. The alternative public key extraction phase is shown in Fig. 3.5.

The security proof for this DKG protocol with alternative key extraction phase is very similar to the previous proof. Instead of giving the full proof, we highlight the differences:

- The simulator sets the $g^{x_i^*}$ for the non-corrupted devices such that the public key will be y. For the zero knowledge proof to hold, SIM chooses a random $d^* \in_R \mathbb{Z}_\ell^*$ and forces the outcome of the open reconstruction of the challenge to d^*Q. For each non-corrupted $\mathcal{D}_i$, SIM computes the commitments $\beta_i^* = g^{z_i}(g^{x_i^*})^{-\tilde{d}}$ and $B_i^* = z_iS_i - \tilde{d}x_iS_i$, for random $z_i \in_R \mathbb{Z}_\ell^*$ and $\tilde{d} = \psi(d^*Q)$.
  (a) SIM broadcasts $\langle g^{x_i^*}, s_iP'', \beta_i^*, B_i^* \rangle$ for each non-corrupted $\mathcal{D}_i$.
  (b) All devices run the Joint PVSS and hold shares d_iQ and d'_iQ. SIM forces the outcome of the open reconstruction of the challenge to d^*Q.
    – SIM computes the d_jQ from the corrupted devices by interpolation of $t+1$ shares d_iQ of the non-corrupted devices.
    – SIM sets the d_i^*Q for the non-corrupted devices by interpolation of the d_jQ of the corrupted devices and d^*Q.
    – By knowing η, SIM will compute $d_i'^*Q$ such that $d_iQ + \eta d'_iQ = d_i^*Q + \eta d_i'^*Q$. As such, the broadcasted shares d_i^*Q, $d_i'^*Q$, will verify against the commitments.
  (c) All $Z_i^* = z_iQ$ are broadcasted and correctly verified.
  (d) At the end of the protocol the public key is computed as the given y.

The qualified devices expose g^{x_i} to compute the public key $y = g^x$.

(a) Each $\mathcal{D}_i$ in QUAL broadcasts $g^{x_i} = \hat{e}(s_i^{-1}P, C_i)$ and s_iP''. It is easily verified that $\hat{e}(s_iP'', Q) = \hat{e}(P'', S_i)$. In addition, $\mathcal{D}_i$ chooses a random $r_i \in_R \mathbb{Z}_\ell^*$ and broadcasts commitments g^{r_i} and r_iS_i.

(b) Generation of the uniform challenge, needed in the zero-knowledge proof.

- Devices in QUAL run a Joint PVSS and obtain protected shares d_iS_i and d'_iS_i, which are broadcasted and verified. We denote the commitments of this Joint PVSS as $B_{i,k}$.
- Open reconstruction of dQ. Devices in QUAL broadcast d_iQ and d'_iQ. These are verified by checking that

$$\hat{e}(P, d_iQ) \cdot \hat{e}(P', d'_iQ) = \prod_{k=0}^{t} \hat{e}(B_k, Q)^{j^k}$$

$$\text{with} \qquad B_k = \sum_{i \in QUAL} B_{i,k} \, .$$

- dQ is mapped to the challenge $\tilde{d} = \psi(dQ)$, where ψ is a bijective map from $\mathbb{G}_2$ to $\mathbb{Z}_\ell$.

(c) Each $\mathcal{D}_i$ broadcasts $Z_i = s_i^{-1}(r_iS_i + \tilde{d}C_i) = (r_i + \tilde{d}x_i)Q$ and any device can verify that

$$\hat{e}(P, Z_i) = g^{r_i}(g^{x_i})^{\tilde{d}} \quad \text{and} \quad \hat{e}(s_iP'', Z_i) = \hat{e}(P'', r_iS_i) \cdot \hat{e}(P'', C_i)^{\tilde{d}} \, .$$

(d) The public key y is computed from $t+1$ correctly verified $\alpha_i = g^{x_i}$ as

$$y = \prod \alpha_i^{\lambda_i} \, .$$

Figure 3.5: Alternative Public Key Extraction.

- To prevent an adversary from being able to distinguish between a real run of the protocol and a simulation, the output distribution must be identical.

	REAL	SIM
2.	$g^{r_i}, r_i S_i$	β_i^*, B_i^*
3.	$d_i Q, d_i' Q$	$d_i^* Q, d_i'^* Q$

2. The value r_i was chosen uniformly at random. In the simulation $\beta_i^* = g^{z_i}(g^{x_i^*})^{-\tilde{d}}$ and $B_i^* = z_i S_i - \tilde{d} x_i S_i$. The value z_i is uniformly random and $\tilde{d} = \psi(d^* Q)$ is derived from the uniformly random d^*.
3. Since the following relation holds, $d_i Q + \eta d_i' Q = d_i^* Q + \eta d_i'^* Q$, it suffices to show that both $d_i Q$ and $d_i^* Q$ have identical distributions. The value d was chosen uniformly at random.

- By broadcasting the modified $d_i^* Q$, we introduce a new assumption. An adversary cannot distinguish $\langle Q, S_i, d_i Q, d_i S_i \rangle$ from $\langle Q, S_i, d_i^* Q, d_i S_i \rangle$. This is the divisional variant of the DDH assumption, which is a weaker assumption than the coDBDH-2 assumption, meaning that if one could not solve the coDBDH-2 problem, one can also not solve the DDH problem. Knowledge of P allows to calculate g^{d_i} and $g^{d_i^*}$ and to transform this to the divisional variant of the coDBDH-2 assumption.

3.3.3 Extended Distributed Key Generation

Uniformity of the group's private key is already guaranteed by the basic DKG protocol. The aim of this extension is to guarantee the uniformity of each participant's share. A rushing adversary could conduct the attack as described by Gennaro [71][5] for biasing the group's private key, to influence the value of one correction factor. A worst case adversary, having corrupted t participants, gains an advantage towards breaking the scheme, by knowing t shares and having information about an additional share. We introduce an extra round in the DKG protocol.

Each participant P_i will broadcast additional values B_i, B_i' and an ordered pair $\langle V_i, W_i \rangle$. The first element of the pair can be seen as a vote for either $f_i(z)$ or $f_i'(z)$. The vote determines if $x_{ij} S_j$ or $x_{ij}' S_j$ will be used to determine the correction factor C_j. The vote is unknown to the other participants, since they cannot distinguish the order of the ordered pair. However, the broadcasted values B_i, B_i' commit P_i to his vote. The commitment can be publicly verified. If the verification fails, P_i is not added to the set of qualified participants QUAL.

[5] This attack is briefly discussed in Sect. 3.3.2.

Since the correction factors depend on the votes of the qualified participants, all votes have to be opened. The votes of qualified participants that did not open their votes, can be calculated by any set of $(t + 1)$ honest participants. Each collaborating participant P_i broadcasts helper values for each participant P_j whose vote is not opened yet. The public key is computed as in the original protocol. The protocol is shown in Fig. 3.6.

3.4 Threshold Applications

In this section our construction is used to turn discrete-log schemes into threshold variants with protected shares. It is not our intention to give a rigorous proof of security of these variants. We rather want to demonstrate the ease with which our construction fits into existing schemes. We do this for the ElGamal [66] and the Cramer-Shoup [43] cryptosystems, where we show how pairings allow implicit use of the shares, i.e., without having to reveal them explicitly, and the Schnorr [142] signature scheme.

Devices that combines partial decryptions or signatures to output decryptions or signature are referred to as combining devices. In priciple this can be any personal device, provided it stores or has access to all protected shares.

3.4.1 ElGamal

Basic Scheme. We define the ElGamal [66] scheme in $\mathbb{G}_T$ with some minor modifications; the randomness is moved from $\mathbb{G}_T$ to $\mathbb{G}_1$ and the private key is an element of $\mathbb{G}_2$ instead of $\mathbb{Z}_\ell^*$, i.e., $xQ \in \mathbb{G}_2$ for some $x \in_R \mathbb{Z}_\ell^*$. Let $y = \hat{e}(P, Q)^x$ be the corresponding public key. Encryption and decryption are then defined as follows.

- **Encrypt(*PubPar*,*y*,*m*):** To encrypt a message $m \in \mathbb{G}_T$ under the public key y, choose a random $k \in_R \mathbb{Z}_\ell^*$ and output the ciphertext
$$(R, e) = (kP, my^k) \in \mathbb{G}_1 \times \mathbb{G}_T\,.$$
- **Decrypt(*PubPar*,*xQ*,(R, e)):** To decrypt the given ciphertext (R, e) output the plaintext
$$m = \frac{e}{\hat{e}(R, xQ)} \in \mathbb{G}_T\,.$$

1. All participants run an instance of PVSS protocol. In addition to step 1(a) of the original DKG, each participant P_i commits to a vote b_i for one of his two randomly generated polynomials. This vote is validated in step 1(c). If validation fails P_i is not added to QUAL.

   (a) Participant P_i chooses random values $b_i \in_R \{0,1\}$, $r_i, r'_i \in_R \mathbb{Z}^*_\ell$ and broadcasts $B_i = c_{i,0}Q + r_iQ$, $B'_i = c'_{i,0}Q + r'_iQ$. Then P_i computes r_iP and r'_iP' and broadcasts $\langle V_i, W_i\rangle = \langle r_iP, r'_iP'\rangle$ if $b_i = 1$. Otherwise, if $b_i = 0$, $\langle V_i, W_i\rangle = \langle r'_iP', r_iP\rangle$.

   (c) For each P_i it is verified that

   $$\hat{e}(P, B_i) \cdot \hat{e}(P', B'_i) = \hat{e}(V_i, Q) \cdot \hat{e}(W_i, Q) \cdot \hat{e}(A_{i,0}, Q)\,.$$

2. Before the correction factors can be computed in step 2 of the DKG protocol, the votes have to be opened. Any subset of $t + 1$ honest participants can open the votes so no corrupt party can refuse to open his vote once he has committed to it.

   (a) Each P_i broadcasts b_i and $g^{c_{i,0}}$, and it is verified that

   $$g^{c_{i,0}} \cdot \hat{e}(Z_i, Q) = \hat{e}(P, B_i) \quad \text{with} \quad Z_i = b_iV_i + (1-b_i)W_i\,. \tag{3.3}$$

   (b) Any subset of t honest participants can open the votes of the qualified participants whose votes have not been opened. For any such participant P_j, P_i will broadcast $s_i^{-1}r_iP$ and $r_i^{-1}x_{ji}S_i$. It is verified that

   $$\hat{e}(s_i^{-1}r_iP, S_i) = \hat{e}(V_i, Q) \quad \text{and} \quad \hat{e}(V_i, r_i^{-1}x_{ji}S_i) = \hat{e}(P, x_{ji}S_i)\,.$$

   If the verification succeeds, the helper values allow to compute

   $$g^{c_{j,0}} = \prod \hat{e}(s_i^{-1}r_iP, r_i^{-1}x_{ji}S_i)^{\lambda_i}$$

   and thus to open the vote of P_j by means of (3.3).

   (c) Based on the votes, the correction factors are computed as

   $$C_j = \prod_{\text{QUAL}} b_i\, x_{ij}S_j + (1 - b_i)x'_{ij}S_j\,.$$

3. The public key is computed as in the original protocol.

Figure 3.6: Extended DKG to prevent bias of public correction factors.

Threshold Variant. Encryption in the threshold variant is the same as in the basic scheme. To decrypt a given ciphertext we have to combine the randomness kP with $t+1$ shares x_iQ, which are stored as x_iS_i. If the shares were stored as $g^{x_i s_i}$, it would have been impossible to combine them with the randomiser or the ElGamal encryption and for each device $\mathcal{D}_i$ the ciphertext would contain something like $g^{x_i s_i k}$. By taking advantage of the bilinearity of the pairing, the size of the ciphertext remains constant. Note that $\mathcal{D}_i$ never reveals its share explicitly; its private key is combined with the randomness from the ciphertext and then paired with the correction factor. The cost of providing a partial decryption is minimal, namely one elliptic-curve point multiplication. In this way we can use small devices as partial decryption oracles. The decryption procedure goes as follows.

- **T-Decrypt(**PubPar**,**$\{\mathcal{D}_i\ , S_i\}$**,**(R,e)**):** To decrypt the ciphertext (R,e) each device $\mathcal{D}_i$ provides a partial decryption
$$D_i = s_i^{-1}R = s_i^{-1}kP \in \mathbb{G}_1 \ .$$
The combining device receives the contributions D_i and verifies that $\hat{e}(D_i,S_i) = \hat{e}(R,Q)$. It then combines $t+1$ contributions to output the plaintext
$$m = \frac{e}{d} \quad \text{with} \quad d = \prod \hat{e}(D_i,C_i)^{\lambda_i} \ .$$

3.4.2 Cramer-Shoup

Basic Scheme. Cramer and Shoup [43] presented an ElGamal based cryptosystem in the standard model that provides ciphertext indistinguishability under adaptive chosen ciphertext attacks (IND-CCA2). We define their scheme in $\mathbb{G}_T$ with the same modifications as in the ElGamal scheme; the first two (random) elements in the ciphertext are moved from $\mathbb{G}_T$ to $\mathbb{G}_1$ and the private key is a tuple from $\mathbb{G}_2^5$ instead of $(\mathbb{Z}_\ell^*)^5$. Let $H : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_T \to \mathbb{Z}_\ell$ be an element of a family of universal one-way hash functions. The private key is
$$\mathit{privK} = (x_1Q, x_2Q, y_1Q, y_2Q, zQ) \in_R \mathbb{G}_2^5$$
and the public key is
$$\mathit{pubK} = (c,d,h) = (g_1^{x_1}g_2^{x_2}, g_1^{y_1}g_2^{y_2}, g_1^{z}) \in \mathbb{G}_T^3 \, .$$
Encryption and decryption are defined as follows.

- **Encrypt(**PubPar**,**pubK**,**m**):** To encrypt a message $m \in \mathbb{G}_T$ under pubK, choose a random $k \in_R \mathbb{Z}_\ell$ and output the ciphertext
$$(U_1,U_2,e,v) = (kP, kP', mh^k, c^k d^{k\alpha}) \in \mathbb{G}_1^2 \times \mathbb{G}_T^2 \text{ with } \alpha = H(U_1,U_2,e) \ .$$

- **Decrypt(*PubPar*,*privK*,(U_1, U_2, e, v)):** To decrypt the given ciphertext (U_1, U_2, e, v), first compute $\alpha = H(U_1, U_2, e)$ and validate the ciphertext by testing if

$$\hat{e}(U_1, x_1Q + y_1\alpha Q) \cdot \hat{e}(U_2, x_2Q + y_2\alpha Q) = v\,.$$

If the test fails, the ciphertext is rejected, otherwise output the plaintext

$$m = \frac{e}{\hat{e}(U_1, zQ)} \in \mathbb{G}_T\,.$$

Threshold Variant. It is clear that the Cramer-Shoup public key is not immediately established from running five instances of our DKG protocol. The decomposition of $c = g_1^{x_1}g_2^{x_2}$ and $d = g_1^{y_1}g_2^{y_2}$ should not be known. We can solve this problem by introducing a DKG variant with a third polynomial $f''(z)$ to generate the public keys c and d. The key generation is thereby reduced to two runs of the variant DKG protocol and one run of the basic DKG protocol. This results in five protected shares $C_i^{x_1}, C_i^{x_2}, C_i^{y_1}, C_i^{y_2}$ and C_i^z for each device. We will now describe this variant.

Without loss of generality, we will describe this variant for the generation of the public key $c = g_1^{x_1}g_2^{x_2}$ and the shares $C_i^{x_1}, C_i^{x_2}$. Each device now receives three instead of two shares. The public key is extracted by revealing the third share and by proving the discrete log equality of $g_3^{x''_i}$ and x''_iS_i. The two parts of the private keys that are shared are x_1Q and x_2Q, where $x_1 = \sum_{i\in\text{QUAL}} c_{i,0}$ and $x_2 = \sum_{i\in\text{QUAL}} c'_{i,0}$ are the respective sums of the private inputs $c_{i,0}$ and $c'_{i,0}$ of the qualified dealers $\mathcal{D}_i$. The corresponding part of the public key is $c = g_1^{x_1}g_2^{x_2}$. The details of the protocol are shown in Fig. 3.7.

Input: *PubPar*, the set of participating devices $\mathcal{D}_i$ and their public keys S_i, and the threshold t
Output: Protected shares $C_i = x_iS_i$ and $C'_i = x'_iS_i$, , with x_iQ and x'_iQ shares of $\mathcal{D}_i$, and the group's public key $y = g_1^x g_2^{x'}$

1. All participating devices $\mathcal{D}_i$ run a modified PVSS protocol simultaneously, the subshares are only broadcasted after receiving all commitments from all participating devices. See Fig. 3.8.
2. Extraction of the public key. See Fig. 3.9.

Figure 3.7: DKG with Protected Shares for Cramer-Shoup.

All participating devices $\mathcal{D}_i$ run a modified PVSS protocol simultaneously, the subshares are only broadcasted after receiving all commitments from all participating devices.

(a) Each $\mathcal{D}_i$ constructs three polynomials $f_i(z)$, $f'_i(z)$ and $f''_i(z)$ of degree t by choosing random coefficients $c_{i,k}, c'_{i,k}, c''_{i,k} \in_R \mathbb{Z}^*_\ell$ for $k = 0 \ldots t$:

$$f_i(z) = c_{i,0} + \cdots + c_{i,t} z^t \quad , \quad f'_i(z) = c'_{i,0} + \cdots + c'_{i,t} z^t \quad ,$$

$$f''_i(z) = c''_{i,0} + \cdots + c''_{i,t} z^t$$

and broadcasts commitments

$$A_{i,k} = c_{i,k} P + c'_{i,k} P' + c''_{i,k} P'' \quad \text{for} \quad k = 0 \ldots t \,.$$

(b) For each device $\mathcal{D}_j$, each $\mathcal{D}_i$ computes and broadcasts

$$x_{ij} S_j \ , \ x'_{ij} S_j \ , \ x''_{ij} S_j \quad \text{with} \quad x_{ij} = f_i(j) \ , \ x'_{ij} = f'_i(j) \ , \ x''_{ij} = f''_i(j) \,.$$

(c) Each device verifies the broadcasted shares for all $\mathcal{D}_i$ by checking that

$$\hat{e}(P, x_{ij} S_j) \cdot \hat{e}(P', x'_{ij} S_j) \cdot \hat{e}(P'', x''_{ij} S_j) = \prod_{k=0}^{t} \hat{e}(A_{i,k}, S_j)^{j^k} \,.$$

Each $\mathcal{D}_i$ that performed a valid PVSS is added to the list of qualified devices, denoted by QUAL. For each $\mathcal{D}_i$ the public correction factors are computed

$$C_i = x_i S_i = \sum_{j \in \text{QUAL}} x_{ji} S_i \quad , \quad C'_i = x'_i S_i = \sum_{j \in \text{QUAL}} x'_{ji} S_i \,.$$

Figure 3.8: Distributed Generation of Protected Shares for Cramer-Shoup.

Extraction of the public key $y = g_1^x g_2^{x'}$.

(a) Each $\mathcal{D}_i$ in QUAL chooses a random $r_i \in_R \mathbb{Z}_\ell^*$ and broadcasts $x_i''Q$, $s_i r_i P''$ and $r_i P''$. It is easily verified that

$$\hat{e}(s_i r_i P'', Q) = \hat{e}(r_i P'', S_i) \qquad \text{and}$$

$$\hat{e}(s_i r_i P'', x_i'' Q) = \hat{e}(r_i P'', \sum_{j \in \mathrm{QUAL}} x_{ji}'' S_i) \ .$$

(b) The public key y is computed from $t+1$ correctly verified $x_i''Q$, as

$$y = g_1^x g_2^{x'} = \frac{\hat{e}(A_0, Q)}{\prod \hat{e}(P'', x_i''Q)^{\lambda_i}} \quad \text{with} \quad A_0 = \sum_{j \in QUAL} A_{j,0} \, .$$

Figure 3.9: Public Key Extraction for Cramer-Shoup.

Encryption is the same as in the basic scheme. The decryption routine, which applies the same ideas as in the threshold ElGamal scheme, goes as follows. Note that the cost of providing a partial decryption is minimal, namely two elliptic-curve point multiplications.

- **T-Decrypt(**$PubPar$**,**$\{\mathcal{D}_i, S_i\}$**,**(U_1, U_2, e, v)**):** To decrypt the given ciphertext (U_1, U_2, e, v) each device $\mathcal{D}_i$ provides $D_i = s_i^{-1}U_1$ and $D_i' = s_i^{-1}U_2$. The combining device verifies that $\hat{e}(D_i, S_i) = \hat{e}(U_1, Q)$ and $\hat{e}(D_i', S_i) = \hat{e}(U_2, Q)$. It then computes

$$v_i = \hat{e}(D_i, C_i^{x_1} + \alpha C_i^{y_1}) \cdot \hat{e}(D_i', C_i^{x_2} + \alpha C_i^{y_2}) \, .$$

  and combines $t+1$ values v_i to validate the ciphertext by testing that $v = \prod v_i^{\lambda_i}$. If validation fails, the ciphertext is rejected. The combining device combines $t+1$ contributions to output the plaintext

$$m = \frac{e}{d} \quad \text{with} \quad d = \prod \hat{e}(D_i, C_i^z)^{\lambda_i} \, .$$

3.4.3 Schnorr Signatures

The Schnorr signature scheme [142] is an example of a scheme that provides existential unforgeability under an adaptive chosen-message attack in the random oracle model [135] and has been used many times to create a threshold signature scheme, e.g., in [71, 3]. We will define the signature scheme in $\mathbb{G}_T$ and then extend it to a threshold variant.

Basic Scheme. Let $H' : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_\ell$ be a cryptographic hash function. Let the private key be $xQ \in \mathbb{G}_2$ for some $x \in_R \mathbb{Z}_\ell^*$ and let $y = g^x \in \mathbb{G}_T$ be the public key.

- **Sign(*PubPar*,xQ,m):** To sign a message $m \in \{0,1\}^*$ with the private key xQ choose a random $k \in_R \mathbb{Z}_\ell$, compute $r = \hat{e}(P, kQ)$ and $c = H'(m, r)$, and output the signature

$$(c, \sigma) = (H'(m, r), kQ + c\,xQ) \in \mathbb{Z}_\ell \times \mathbb{G}_2\,.$$

- **Verify(*PubPar*,y,(c,σ),m):** To verify the signature (c, σ) on a message m compute $\tilde{r} = \hat{e}(P, \sigma) y^{-c}$ and verify equality of $c = H'(m, \tilde{r})$.

Threshold Variant. The basic scheme naturally extends to a threshold variant. As opposed to the encryption schemes, the bilinearity of the pairing is not really needed. However, the signing devices need to share some randomness and will, therefore, run the DKG protocol of Sect. 3.3.2. Signature verification is the same as in the basic scheme. Signing goes as follows.

- **T-Sign(*PubPar*,$\{\mathcal{D}_i\}$,m):** To sign a message $m \in \{0,1\}^*$ with the group's private key the devices $\mathcal{D}_i$ will run an instance of the DKG protocol of Sect. 3.3.2. Each device then holds a share $k_i S_i$ in protected form of $kQ \in \mathbb{G}_2$. Because the value $r = \hat{e}(P, kQ) \in \mathbb{G}_T$ is publicly computed at the end of the protocol, each device can compute $c = H'(m, r)$ and $\sigma_i = s_i^{-1}(k_i S_i + cC_i) = (k_i + c\,x_i)Q$, which is sent to the combining device. Note that these partial signatures can be verified since the output of the DKG protocols contained g^{k_i} and g^{x_i}. Values that were not in the output can be computed through interpolation. The combining device computes the signing equation $\sigma = \sum \sigma_i \lambda_i$ and outputs the signature

$$(c, \sigma) = (H'(m, r), kQ + c\,xQ) \in \mathbb{Z}_\ell \times \mathbb{G}_2\,.$$

3.5 Conclusion

In this chapter, we have shown how to increase resilience when deploying threshold cryptography on things that think by including small devices with limited or no secure writeable storage capabilities. Assuming these devices have some support for public-key functionality and stored unique private key, shares can be stored in protected form. By using bilinear pairings, this particular form yields some advantages. The most important feature is public verifiability, which makes explicit private channels and cumbersome complaint procedures obsolete. Moreover, not all devices need to be present during group setup. The bilinearity of the pairings also leads to constant size ciphertexts, independant of the devices part of the threshold scheme.

We have demonstrated how to adopt the protected shares in existing discrete-log based signature schemes and cryptosystems. Because shares are never needed in unprotected form, small devices can be used as decryption oracles at a minimal cost.

Chapter 4

Proactive Threshold Cryptography

Security can be enhanced without changing the secret, by distributed generation of new shares for the participants. Furthermore, by handing out new shares, the set of devices can be altered, i.e. devices can be added or removed. In this chapter we deal with devices that are not present during the share update process and the authorisation for this process. Device management is user-centric, hence the usability of the proposed authorisation protocol is evaluated.

4.1 Introduction

Threshold cryptography can be made more robust by making it proactive. Proactive security allows the overall system to recover from partial compromises, for example, a device's share that is compromised by the adversary while the device is left unattended (lunch time attack). With the periodic updates of the devices' shares, each device gets a new share and erases its old one. Given the dynamic ability of the Shamir secret sharing scheme (see Sect. 2.3.1), the shares that the adversary already collected are rendered useless. Hence, the adversary only has a limited time frame in which it needs to compromise $t + 1$ devices to break security of the threshold system. Several Proactive Secret Sharing (PSS) protocols [58, 60, 61, 86, 87, 136] have been proposed. However, these protocols only allow to refresh the shares within the existing threshold structure, i.e. the set of shareholders and the threshold number are fixed.

Secret Redistribution or Resharing protocols [52, 57, 172] also allow to move from a $(t+1,n)$-secret sharing to a $(t'+1,n')$-secret sharing. Hence, resharing protocols not only allow to change the threshold number, but also the number of devices. By running an instance of a resharing protocol, devices can be added or removed from the set of personal devices. Thus, the end-user can discard broken devices, use new devices or even sell devices without reducing the overall security level.

Even though good protocols exist to achieve these goals, before deploying these protocols on personal devices, two aspects need to be explored further in detail. First, we investigate how to deal with absent devices. Second, we study the following question: How will the end-user authorise his personal devices to run an instance of the reshare protocol, taking into account the new set of personal devices and threshold?

4.2 Resharing

Resharing protocols are very similar to distributed key generation protocols. For both, each player acts as a dealer in a verifiable secret sharing scheme. Upon successfully completing the protocol, each device holds a share that is equal to the sum of the subshares sent to that device. The objective of a resharing protocol is to distribute new threshold shares of the existing private key, while the objective of a DKG protocol is the joint generation of a threshold shared private key and the corresponding public key. The difference between resharing and DKG protocols is that for resharing, instead of each party sharing a random value, the current share is shared with the other parties.

The private key does not change during a resharing protocol. The free term of the polynomial containing all shares does not change. This means that compared to a DKG protocol, no special precautions need to be taken to make sure that the private key is uniformly distributed[1]. Hence, we can use the resharing protocol as proposed by Wong *et al.* [172] (see Fig. 4.1), for which the commitments to the subshares are not perfectly hiding (instead, the commitments are only computationally hiding). These perfectly binding commitments allow for efficient verification. To ensure correctness, i.e. the new shares to be valid, the old shares (cf. Eqn. (4.1)) and the subshares (cf. Eqn. (4.2)) need to be valid. Please note that, unlike in the previous chapter, the shares need to be kept secret and pair-wise private channels between the devices are necessary. These private channels can be constructed using encryption over

[1]The need to guarantee a uniformly distributed key during a distributed key generation protocol, was pointed out by Gennaro [71].

the broadcast channel. As a consequence, all devices need to be present at each instance of the resharing protocol. In the next section, this constraint is relaxed. We propose some strategies to deal with absent devices.

Given the public key $y = g^x$, to reshare the secret x from a $(t+1, n)$ to a $(t'+1, n')$ scheme:

1. Each device $\mathcal{D}_i$ in the set S, a subset of size greater than t devices that are part of the original scheme, constructs one polynomial $f_i(z)$ of degree t' by choosing random coefficients $a_{i,k} \in_R \mathbb{Z}_q^*$ for $k = 1 \dots t'$ and setting $a_{i,0}$ to its old share s_i:

$$f_i(z) = a_{i,0} + \cdots + a_{i,t'} z^{t'}$$

   and broadcasts commitments (for g a generator of subgroup of $\mathbb{Z}_p$ with order q)

$$b_{i,k} = g^{a_{i,k}} \bmod p \quad \text{for} \quad k = 0 \dots t' .$$

2. For each device $\mathcal{D}_j$ in the new scheme, each $\mathcal{D}_i$ computes and sends over a private channel

$$s_{ij} = f_i(j) \bmod q .$$

3. Each device $\mathcal{D}_j$ verifies that

$$y = \prod_{i \in S} (b_{i,0})^{\lambda_i} \bmod p \tag{4.1}$$

   and

$$g^{s_{ij}} = \prod_{k=0}^{t'} (b_{i,k})^{j^k} \bmod p \qquad \forall\, i \in S . \tag{4.2}$$

   If the verification succeeds a "commit" is broadcast.

4. If all $\mathcal{D}_j$ agree to commit, the new shares are generated and stored

$$s'_j = \sum_{i \in S} s_{i,j} \lambda_i \bmod q .$$

Figure 4.1: Resharing Protocol Proposed by Wong *et al.* [172].

Let τ denote the number of devices that are compromised by the adversary. Let $\tau_a \leq \tau$ denote the number of actively compromised devices. These devices are controlled by the adversary during the resharing protocol and will remain compromised after resharing. All other (not actively) compromised devices will be recovered, the adversary will not know their new shares. The number of compromised devices after resharing is τ' and is equal to τ_a if the threshold sharing structure is not changed. In this case, the adversary does not gain from resharing.

Adding or Removing Devices. When allowing a user to add or remove devices while resharing we introduce a possible advantage for the adversary. Let a denote the number of added devices and r the number of removed devices. The number of devices after resharing is $n' = n + a - r$. When removing an uncompromised device, the total number of devices decreases while the number of actively compromised devices remains the same. When adding an actively compromised device, both the total number of devices and the total number of actively compromised devices increase. For a worst case adversary, the number of compromised devices after resharing is $\tau' = \tau_a + a$.

One can calculate bounds on the number of devices the adversary can actively compromise, preventing the adversary from learning the shared private key (Eqn. (2.1)) and mounting a Denial of Service attack (Eqn. (2.2)):

$$\tau_a < t' + 1 - a \qquad\qquad \tau_a < n - r - t' \,.$$

We assume that the threshold is set automatically $t' + 1 = \lceil \frac{n'}{2} \rceil$ to maintain the highest possible security while the scheme remains functional (see Sect. 2.4.2). This results in the following bound:

$$\tau_a < \left\lceil \frac{n - a - r}{2} \right\rceil \,.$$

For the strongest security guarantees, it is assumed that the number of actively compromised devices, τ_a, is as close as possible to the maximum number of devices the adversary is allowed to compromise, τ_{max}. Hence, the end-user can only add or remove one device at the time.

$$\begin{array}{lll} \tau_a & \leq & \tau_{max} \quad \text{for } n \text{ even} \\ & < & \tau_{max} \quad \text{for } n \text{ odd} \;. \end{array}$$

The difference between the cases for the number of devices before resharing, n, to be even or odd, is a direct consequence of the way the threshold number is set. For an even number of devices, τ_a can be equal τ_{max}, meaning that the adversary gains nothing by resharing. Adding one device results in an increase of the threshold number, removing one device does not change it. For an odd number of devices, τ_a must be smaller than τ_{max}, since adding a device has no effect on the threshold number, while removing one device lowers it.

4.3 Absent Parties

Although the resharing protocol proposed by Wong *et al.* [172] does not require all the devices in the current scheme to be present (any honest subset of size $t+1$ or bigger is sufficient), it requires all devices in the new scheme to be present and participating in the verification. However, these personal mobile devices are not always available, for example, empty batteries or devices left at home. Given this constraint, we still want to be able to reshare without excluding the absent devices from the threshold secret sharing. In this section, we will only discuss the case where the set of devices and threshold number do not change during resharing. However, it is possible to alter the threshold number, add or remove devices. In the following, we assume that only $m > t$ out of n devices are present.

An obvious solution would be to run an instance of the reshare protocol from a $(t+1, n)$ to a $(t'+1, m)$ scheme and at a later point in time, when all n devices are present, run another instance going from a $(t'+1, m)$ to a $(t+1, n)$ scheme. This solution is not desirable, first of all since it requires at least two (all n devices might never be available at the same time) instances of the resharing protocol. Second there is the trade-off between security and reliability. By changing the threshold $t'+1$ dynamically with the number of available devices m, the overall security is lowered since the adversary can break the security of the scheme by compromising a smaller number than the original threshold number of devices. When fixing the threshold $t'+1 = t+1$, the reliability of the scheme is affected since only $m-t-1$ instead of $n-t-1$ devices can be absent. Even worse, since the adversary is allowed to compromise t devices, he can perform a permanent denial of service attack if $m \leq 2t$.

Two more promising approaches are to make sure the absent devices get their new shares afterwards or to fix the shares of the absent devices. These will each be discussed in the following sections.

4.3.1 Hand Out Shares Afterwards

A resharing protocol is run among all parties and the shares of the absent devices are stored, such that they can be sent later on when these devices are available. Verification of the subshares for the absent devices implies public verifiability. Unfortunately, we cannot reuse the public verifiability techniques that were introduced in the previous chapter. Recall that we constructed shares in a protected form. However these protected shares are publicly available. In such a setting, one cannot assume that shares are erased and thus the adversary is not restricted to a limited time frame for compromising the threshold number of devices. Moreover, it is impossible to remove devices in that setting. Resharing requires private channels.

We take a closer look at the private channels between devices. To construct private channels over the broadcast channel, we use encryption. Public verifiability can be achieved by carefully selecting an appropriate cryptosystem, in this case a homomorphic one. The Paillier cryptosystem [123], later extended by Damgård and Jurick [45, 97], allows us to compute the encryption of the sum of two plaintexts: $E(m_1, r_1) \cdot E(m_2, r_2) = E(m_1 + m_2, r_1 r_2)$. Each device $\mathcal{D}_j$ has a public/private Paillier key pair of the form $(\{N_j, g_j\}, d_j)$.

Each dealer $\mathcal{D}_i$ will commit to the coefficients of its polynomial $b_{i,k}$ and broadcast subshares encrypted (under the recipients' public keys) to the other devices. The encrypted subshare for device $\mathcal{D}_j$ is $e_{ij} = E_j(s_{ij}, r_{ij}) = (g_j)^{s_{ij}} (r_{ij})^{N_j} \bmod N_j^2$. The dealer has to prove that the encrypted value corresponds to the value that is committed to $c_{ij} = g^{s_{ij}} = \sum_{k=0}^{t} b_{i,k}^{j^k} \bmod p$. The sigma protocol, depicted in Fig. 4.2, allows to prove this.

Given $e = E(s, r), c = C(s)$ and the private input (s, r) of the prover:

1. The prover chooses s', r' and broadcasts
$$e' = E(s', r') \qquad \text{and} \qquad c' = C(s) \ .$$
2. The verifier broadcasts the challenge $h \in \{0, 1\}^l$.
3. The prover broadcasts
$$s'' = s' + hs \bmod q \qquad \text{and} \qquad r'' = r'(r)^h \bmod q \ .$$
4. The verifier verifies
$$e'(e)^h = E(s'', r'') \qquad \text{and} \qquad c'(c)^h = C(s'') \, .$$

Figure 4.2: Sigma Protocol.

Theorem 4. *The proposed sigma protocol is complete and sound.*

Completeness.

- $e'(e)^h = E(s', r')E(s, r)^h = E(s' + hs, r'(r)^h) = E(s'', r'')$.
- $c'(c)^h = C(s')C(s)^h = C(s' + hs) = C(s'')$.

□

Special soundness. From any pair of accepting conversations (e', c', h, s'', r''), $(e', c', \bar{h}, \bar{s''}, \bar{r''})$, we can compute s, r such that $e = E(s, r)$ and $c = C(s)$.

$$e^{h-\bar{h}} = g_j^{s''-\bar{s''}} \left(\frac{r''}{\bar{r''}}\right)^{N_j} \bmod N_j^2 \ .$$

For 2^l smaller than the smallest prime factor of N_j, $(h - \bar{h})$ is co-prime with N_j. Euclid's algorithm[2] allows to find α, β such that $\alpha(h - \bar{h}) + \beta N_j = 1$.

$$e = g_j^{(s''-\bar{s''})\alpha} \left(\left(\frac{r''}{\bar{r''}}\right)^{\alpha} e^{\beta}\right)^{N_j} \bmod N_j^2 \ .$$

$$c^{h-\bar{h}} = g^{s''-\bar{s''}} \bmod p \ .$$

Since the order of the subgroup of the exponent q is known, $(h - \bar{h})^{-1} \bmod q$ can be calculated.

$$c = g^{\frac{s''-\bar{s''}}{h-\bar{h}}} \bmod p \ .$$

$(h - \bar{h})$ divides $(s'' - \bar{s''})$

$$\begin{aligned}
e &= g_j^{\frac{s''-\bar{s''}}{h-\bar{h}}\alpha(h-\bar{h})} \left(\left(\frac{r''}{\bar{r''}}\right)^{\alpha} e^{\beta}\right)^{N_j} \bmod N_j^2 \\
&= g_j^{\frac{s''-\bar{s''}}{h-\bar{h}}(1-\beta N_j)} \left(\left(\frac{r''}{\bar{r''}}\right)^{\alpha} e^{\beta}\right)^{N_j} \bmod N_j^2 \\
&= g_j^{\frac{s''-\bar{s''}}{h-\bar{h}}} \left(\left(\frac{r''}{\bar{r''}}\right)^{\alpha} \left(e g_j^{\frac{\bar{s''}-s''}{h-\bar{h}}}\right)^{\beta}\right)^{N_j} \bmod N_j^2 \\
&= E_j\left(\frac{s''-\bar{s''}}{h-\bar{h}}, \left(\frac{r''}{\bar{r''}}\right)^{\alpha} \left(e g_j^{\frac{\bar{s''}-s''}{h-\bar{h}}}\right)^{\beta}\right) \ .
\end{aligned}$$

[2]Euclid's algorithm is described in book VII and X of Elements (circa 300 BC), written by the Greek mathematician Euclid, making it one of the oldest numerical algorithms still commonly in use.

We found the values s and r

$$s = \frac{s'' - \bar{s''}}{h - \bar{h}} \bmod q \qquad r = \left(\frac{r''}{\bar{r''}}\right)^{\alpha} \left(e g_j^{\frac{\bar{s''} - s''}{h - \bar{h}}}\right)^{\beta} \bmod q\,.$$

$\square$

This sigma protocol can be transformed into a non-interactive proof by applying the Fiat-Shamir heuristic [55]. Let $H()$ be a cryptographic hash function with output length l. The challenge h is replaced by $H(e, e', c, c')$.

Each subshare can be publicly verified. The encrypted subshares for the absent devices can be added to their encrypted new shares and be stored by the present devices. The resulting encrypted share is signed by the present parties, indicating to the absent device that the encrypted new share was verified and deemed correct by at least one honest party. To prevent the reuse of previously signed values, the message to be signed is extended with the commitment to the present share of the intended device. Note that this commitment can be computed as in Eqn. (4.3). At a later time, the absent device gets its encrypted new share and the signatures. This device has to compute the commitment to its present share and verify the signatures. If the signatures verify, the device decrypts its new share and updates its share. To compute and verify these signatures, each device needs an additional public/private (verification/signature) keypair for signing purposes. Alternatively, if the devices share a signature key (for which the corresponding verification key is known), one threshold signature can be placed on the message. In this case, absent devices only need to verify one signature.

4.3.2 Fix Shares

The shares of the absent devices can be fixed, meaning that the absent device's current share equals its new share after resharing took place. This is only possible as long as the number of absent devices is smaller than the degree of the polynomial $n - m < t$ (to have at least one degree of freedom) and the number of present devices is at least the threshold $m \geq t + 1$. For each of the m present devices $\mathcal{D}_i$ in the set S, for each absent device $\mathcal{D}_j$, the following constraint on the polynomial is added (on top of $a_{i,0} = s_i$):

$$f_i(j) = s_i \frac{\lambda_i(j)}{\lambda_i(0)} = s_i \prod_{k \in S, k \neq i} \frac{k - j}{k}\,.$$

To verify the correctness, we also need to make sure that the shares of the absent devices did not change. The following additional equations need to be verified (on top of Eqn. (4.1) and (4.2)):

$$(b_{i,0})^{\frac{\lambda_i(j)}{\lambda_i}} = \prod_{k \in S} (b_{k,0})^{\lambda_k(j)} = g^{s_j} \qquad \forall\, i \in S, j \in \{1, \ldots, n\}/S\,. \tag{4.3}$$

Following this approach, a limited number of devices with publicly available protected shares (see previous chapter) can be used along devices that securely store their own shares in unprotected form. The publicly available protected shares can obviously not be updated and will remain fixed. However, devices that cannot store their shares can be removed in this hybrid setting.

4.4 Authorisation

Little attention has been paid to the problem of authorisation for resharing. Proper authorisation is necessary to prevent an adversary from altering the set of devices part of the threshold secret sharing in such a way that he would be able to break the scheme or mount a Denial of Service attack.

Due to the specific setting, a group of at least $t + 1$ agreeing devices can act as a trusted entity, that decides if resharing is authorised or not. Hence, authentication for resharing is related to voting algorithms in distributed systems, such as the algorithms proposed by Castro [38] and Hardekopf [80].

To achieve user authorisation, the end user has to transfer data between the devices. Hence the user forms an authenticated out-of-band-channel. This is similar to the setting of pairing protocols, for which solutions exist. Pairing protocols allow two devices to securely exchange keys that can be used later on to establish secure communication. Germann *et al.* [67] proposed MANual Authentication (MANA) protocols, Laur *et al.* [104] presented to exchange Short Authenticated Strings (SAS) between two devices. Saxena *et al.* [141] discussed a method for pairing two devices in a more secure and user-friendly way by using an auxiliary third device. Laur *et al.* [105], Nguyen *et al.* [121] and Wang *et al.* [164] also proposed group manual authentication protocols. Gehrmann *et al.* [68] introduced the concept of a Personal Certification Authority. This can reduce the number of MANA protocols that need be carried out by devices with limited storage capabilities. Devices with limited storage capabilities cannot store the public keys of the other devices in the threshold secret sharing. After performing a MANA protocol, a personal certificate, containing the public key of the other device, is constructed. This certificate is then broadcasted and stored by devices with greater storage capabilities.

The end-user plays a critical role and has already been named the “weakest link” in security systems [170]. With the arrival of the ubiquitous society, users are less and less aware of the underlying technologies. Furthermore, the responsibility of protecting personal information is shifting towards the end user, for whom convenience is often more important than security. A usable security scheme will prevent the user from resorting to workarounds or short-cuts that render the system vulnerable. Usability principles for designing secure authentication systems have been continuously gaining importance. It is generally acknowledged that security systems offering a poor user experience result in a substantial gap between the theoretical and actual levels of protection.

We consider the following problem: design a secure and user-friendly [4] protocol to authorise resharing. We solve this problem by introducing two ways to authorise the resharing process: automatic and manual.

The automatic authorisation for resharing requires no user interventions and leaves the user free of any cognitive workload, respecting the principle of psychological acceptability [140] which also states that users must be able to routinely and automatically apply protection mechanisms.

Manual authorisation provides the user with the flexibility needed to alter the set of devices that participate in the threshold secret sharing, e.g., to add or remove a device. It avoids confronting the user with the manual input of passwords, steering away from all related trade-offs between memorability and security [4]. Nevertheless, the manual authorisation process still entails a great deal of user interaction bringing along all the possible pitfalls.

The usability of our proposed protocol is evaluated. The main goal of this evaluation is to uncover interaction flaws, but it is also meant to uncover real security defects.

4.4.1 Automatic

Shares of the private key of each device will be periodically updated. Since no reliable global clock exists within the system, the number of performed decryptions/signatures will be used as an approximation for time. Each device has a local counter, counting the number of performed decryptions/signatures since the last (re)sharing of the private key. We define c as the number of decryptions/signatures after which resharing is recommended. The number of decryptions/signatures after which resharing is necessary is $C > c$. After successful resharing all devices reset their local counter $c = 0$.

If the local counter of one device reaches c, this device requests automatic authorisation for resharing. If this request is denied, the device augments its local counter with a step Δ, $c = c + \Delta$. If the local counter of one device reaches C, several attempts to get automatic authorisation for resharing failed. However, at this moment resharing is necessary. In this case the user is alerted by his devices that have a Graphic User Interface (GUI) that manual authorisation for resharing is needed.

Resharing is only authorised if all n devices are present[3] and at least $t + 1$ devices agree to reshare. A device agrees to reshare if its local counter is equal to or greater than c', defined as follows:

$$c' = \left\lfloor \frac{t}{n-1} c \right\rfloor .$$

This value is based on two assumptions. First, that for each decryption/signature generation only the minimal number of devices co-operates. Second, that all devices, except the one that triggered the automatic resharing protocol, participated in an equal number of decryptions/signature generations. The choice of c' is optimal to stop DoS attacks. The adversary, with the goal of draining the mobile devices' batteries, will not succeed in getting the devices to constantly reshare. Furthermore he cannot stop automatic resharing from taking place.

Replay of the request and/or granting of the request for automatic authorisation needs to be prevented. This is prevented by the introduction of nonces. Figure 4.3 shows the protocol for automatic authorisation.

Instead of each device placing a signature with its private key, a threshold signature protocol can be deployed. Each device puts a partial signature on the request; these partial signatures are then combined to construct the signature on the request. When using a threshold signature only one signature needs to be verified. However, in the case of dishonest parties this can result in many combinations and verifications.

[3]When deploying mechanisms to deal with absent parties (as discussed in previous section), this requirement can be relaxed.

1. Device $\mathcal{D}_j$ $(c_j > c)$ broadcasts a session identifier $sess_{id}$.
2. Each device $\mathcal{D}_i$ broadcasts a nonce $nonce_i$.
3. After receiving all other $(n-1)$ nonces, each device $\mathcal{D}_i$, if its local counter $c_i > c'$, constructs the request for automatic authorisation and broadcasts its approval.

   $$SIGN_i(request) \qquad request = (sess_{id}, nonce_1, \ldots, nonce_n)\ .$$

   If after a certain time a device did not receive all other $(n-1)$ nonces, this device aborts. Device $\mathcal{D}_j$ updates $c = c + \Delta$ when it aborts.
4. After verifying the signatures of t other approvals, each device that has given its approval concludes that automatic resharing is authorised.

Figure 4.3: Protocol for Automatic Authorisation.

4.4.2 Manual

Resharing can be authorised by the user, possibly altering the set of participating devices, e.g. adding or removing one device. The user can also be requested by GUI-enabled devices to manually authorise resharing. This will be the case if the local counter of one device reaches C or the resharing protocol failed.

We distinguish three types of devices: participating, non-participating and new devices. Participating devices refer to devices that are part of the threshold secret sharing both before and after resharing. Devices no longer part of threshold secret sharing after resharing are referred to as non-participating devices; these are the devices that will be removed from the threshold secret sharing. New devices are only part of the threshold secret sharing after resharing; these are the devices that will be added to the threshold secret sharing. The proxy refers to the first device the user interacts with by entering his request for manual authorisation for resharing.

The problem of the user authenticating his request is similar to the "What You See Is What You Sign" problem [103], also known as the "Trusted Path Problem". The user cannot be sure that what is displayed on the screen (his request) is what the device broadcasts on his behalf. The adversary, when controlling the device, could display another request than the one the user authenticates. Furthermore, the adversary should not be able to reproduce and/or replay the user's authentication of a request. The user can manifest himself towards his devices by interaction with the threshold number of these. This allows us to detect cheating devices.

Since we require explicit interaction of the user before resharing takes place, we do not need to put safeguards in place to prevent DoS attacks, as is the case with automatic resharing.

The proposed solution consists of two steps: first the user enters his request at the proxy which broadcasts the request; and second he confirms his request at other devices. Figure 4.4 shows the protocol for manual authorisation. Adding a device is a special case and is discussed afterwards.

1. The user selects device $\mathcal{D}_j$ as the proxy and enters his request. The proxy broadcasts the user's request:

   $$\text{request} = (sess_{id}, \text{ID}_{proxy}, \text{ID}_i, \text{ID}_j, \ldots) \qquad \text{SIGN}_{proxy}(\text{request}) \quad .$$

2. a) Each device verifies that it is a participating device and verifies the approval from the proxy. If correct, the request is displayed for the user to confirm in a user-friendly way.

   b) When the user approves the request at device $\mathcal{D}_i$ the request is signed by this device and the approval is broadcast:

   $$\text{SIGN}_i(\text{request}) \quad .$$

3. After verifying the signatures of t other approvals, each device that has broadcasted the user's approval concludes that manual resharing is authorised and the protocol for resharing can be executed.

Figure 4.4: Protocol for Manual Authorisation.

The user utilises one of his GUI-enabled devices as a proxy to get a list of devices in the neighbourhood. From this list of devices he can select the devices that will participate in the resharing. The proxy broadcasts the user's request to authorise resharing. The request consists of the identities of the participating devices. The request is signed by the proxy with its own signature key.

Each device checks if it is a participating device and verifies the signature on the request. If the signature on the request is correct, the request is displayed for the user to confirm. When a user confirms the request on a device, the request is signed by this device and this signature is broadcast. The user confirms his request on at least t participating devices; in total the user needs to interact with at least $t + 1$ participating devices. Before confirming, a user can verify his request at all GUI-enabled devices. Verifying his request before confirming reduces the adversary's chance of tampering with the request, as discussed later on.

Each participating device verifies the correctness of the signatures and checks whether the request is signed by a participating device. After $t + 1$ valid signatures from participating devices, resharing is authorised and takes place.

Adding a device

When adding a new device, this device is not yet known to the participating devices. The user needs to verify that the device he intends to add is the device that will be added, hence this device first needs to be authenticated to the group of participating devices and vice versa. For this goal we need a way of authenticating new information, in this case public keys. We considered three possible approaches:

1. We only need a manual authentication protocol between the new device and this trusted entity. When performing standard manual authentication protocols between two parties, at least $t + 1$ instances of this protocol are required, which is not user-friendly.
2. Another approach could be to only perform a manual authentication protocol between the new device and the proxy and add a verification step during the resharing protocol. It is only at this point that the user can verify if the device he intended to add is the newly added device. The user is requested to check if the intended device indicates resharing in progress. After confirming this on $t + 1$ of the participating devices, the participating devices commit to the resharing and the key shares are overwritten. The additional verification step puts a burden on the user.
3. Group manual authentication protocols, allow authentication to all devices at once and do not require more effort from the user.

To exchange public keys between the new device and the participating devices, we use the *Group Message Authentication* (GMA) protocol that exchanges

Short Authenticated Strings (SAS) by Laur and Pasini [105], shown in Fig. 4.5. The ordered vectors $\mathbf{G}$, $\hat{\mathbf{m}}_i$ and $\hat{\mathbf{r}}_i$ represent respectively all identities ID_j, messages m_j and random values r_j as received by device $\mathcal{D}_i$ from devices $\mathcal{D}_j$. The commitment in the first phase of the protocol commits a device to the final hash value. Because the commitments are only opened in the second phase, no device can force the hash result to a specific value.

1. R1: each participating device $\mathcal{D}_i$ broadcasts a message and a commitment to a random value r_i:

$$m_i \qquad c_i = commit(r_i)\ .$$

2. R2: all devices open their commitments and verify the correctness of the other commitments. **Abort if abnormal behaviour.**

$$r_i\ .$$

3. SAS: each device $\mathcal{D}_i$ calculates the SAS value. The user forms the authenticated channel to compare these SAS values.

$$SAS_i = h((\mathbf{G}, \hat{\mathbf{m}}_i), \hat{\mathbf{r}}_i)\ .$$

Figure 4.5: SAS-GMA Proposed by Laur and Pasini [105]

The new device starts by sending a session identifier, its identity and its public key as the message in the SAS-GMA protocol with the participating devices. The message of the participating devices in the SAS-GMA protocol consists of the session identifier, the public state and their public key. The public state consists of the identities of all participating and new devices, and the threshold $t+1$. In the second round of the SAS-GMA protocol an additional check is required to detect abnormal behaviour. The new device verifies that all received messages contain the same public state. A single SAS value needs to be broadcasted over the authenticated channel: the user needs to compare the displayed values between his mobile devices.

The comparison of this SAS value is combined with the user's consent for adding this device. The new device will display its SAS value, and the participating devices will ask the user if he wants to add a new device displaying the calculated SAS value. When the user confirms at device $\mathcal{D}_i$, this device constructs and signs the request and broadcasts the approval. After $t+1$ valid signatures from participating devices, resharing is authorised and takes place. Figure 4.6 gives an overview of the protocol to manually authorise adding a device.

1. SAS-GMA R1:

   (a) The new device sends the session identifier, its identity and its public key as message.

   $$m_{\text{NEW}} = sess_{id} \,||\, ID_{\text{NEW}} \,||\, PK_{\text{NEW}} \ .$$

   (b) In response, each participating device $\mathcal{D}_i$ sends the session identifier, the public state and its public key as message.

   $$m_i = sess_{id} \,||\, \mathbf{G} \,||\, t{+}1 \,||\, PK_i \qquad \mathbf{G} = \{ID_1, \ldots, ID_n, ID_{\text{NEW}}\} \ .$$

2. SAS-GMA R2: **Abort if abnormal behaviour**, additional check: the new device verifies that all received messages contain the same public state.

3. SAS-GMA SAS: A single SAS message needs to be broadcasted over the authenticated channel. The user is requested by all participating devices if he wants to add the new device and if it displays the same SAS value.

   $$SAS_{\text{NEW}} \ .$$

   When the user approves at participating device $\mathcal{D}_i$, the request is constructed and signed by this device, and the approval is broadcast.

   $$SIGN_i(request) \qquad \text{with} \qquad request = (sess_{id}, G, SAS_i) \ .$$

4. After verifying the signatures of t other approvals, each device that has broadcasted the user's approval concludes that manual resharing is authorised and resharing takes place.

Figure 4.6: Protocol for Manually Authorisation: Adding a Device

Other applications. The protocol for adding a device can also be used to authorise signatures or decryptions. This authorisation can be necessary to place signatures on documents or to decrypt very sensitive data. In either case the proxy has some data, unknown to the other participating devices. This unknown data is a digest of the data to be signed or an encapsulated symmetric key that needs to be decrypted to decrypt the data.

The SAS-GMA protocol provides authentic data exchange between the participating devices. The proxy's message consists of a session identifier, its identity and the data to be signed or decrypted. Other participating devices just send the session identifier as message. In round two of the SAS-GMA protocol the proxy verifies that the session identifier matches. The request consists of the identity of the proxy, the data to be signed or decrypted, the session identifier and the SAS value.

Verifying request

The end-user could pick as proxy a device that has been actively compromised by the adversary. In this case, the adversary can send an alternate request. A user will be unaware of this cheating proxy unless he verifies his request before confirming it on his devices. Other devices with a graphic interface that are under the control of the adversary can display the original request (as entered by the end-user), making it harder for the end-user to detect the cheating proxy.

The probability of an adversary cheating successfully decreases with each device on which a user verifies his request. The factor by which this probability decreases depends on the number of participating devices, $t + 1 \leq p \leq n$. For each device on which the user verifies his request, the adversary's success probability is reduced by a factor f_p, as defined in Eqn. (4.4). This factor takes into account the worst case adversary: one that controls (actively compromises) the maximum number of devices, and having all these devices participate.

$$f_p = \frac{p-1}{(\tau_a)_{max} - 1} \,. \tag{4.4}$$

In order to reduce the adversary's success probability to $0 \leq success \leq 1$, an end-user should verify his request at $Q_p(success)$ randomly chosen devices:

$$Q_p(success) = \lceil -log_{f_p}(success) \rceil \,. \tag{4.5}$$

By verifying his request at one randomly chosen device, cheating behaviour is identified in more than 50% of the cases for typical values of the number of personal devices and allowing only one device to be removed at the time. Table 4.1 gives an overview for these values. The values f_n and Q_n are applicable when all devices participate in the resharing. The number of devices Q_n where a user needs to verify his request, are provided for the case where the adversary's success probability is reduced to 10% or less.

Table 4.1: Security Overview Resharing.

n	$t+1$	max τ	max τ_a	f_n	$Q_n(.1)$
3	2	1	0	-	-
4	2	1	1	∞	1
5	3	2	1	∞	1
6	3	2	2	5	2
7	4	3	2	6	2
8	4	3	3	3.5	2
9	5	4	3	4	2
10	5	4	4	3	3

4.4.3 Usability Evaluation

Automatic authorisation for resharing requires no interaction with the user, hence only manual authorisation will be evaluated. This evaluation investigates how end-users perceive the proposed protocol for authorising resharing on personal devices.

Methodology

First the test subjects need to be introduced to the concept of threshold things that think, before we can evaluate the usability of the more complex concept of resharing. Towards this end, a series of cartoons (see was developed. These cartoons can be found in Appendix A.

To simulate user interactions and the devices' screens, a web-based interface was built. This web-based interface facilitated field studies, as only a laptop was required. In the mock-up interface for the devices, the user is only given a limited number of choices: reshare the secret, remove or add a device. A user can authorise a resharing instance without changing the devices in the threshold secret sharing. He can also authorise a resharing in which one device is added or removed.

Three scenarios are considered to test the usability of the protocol:

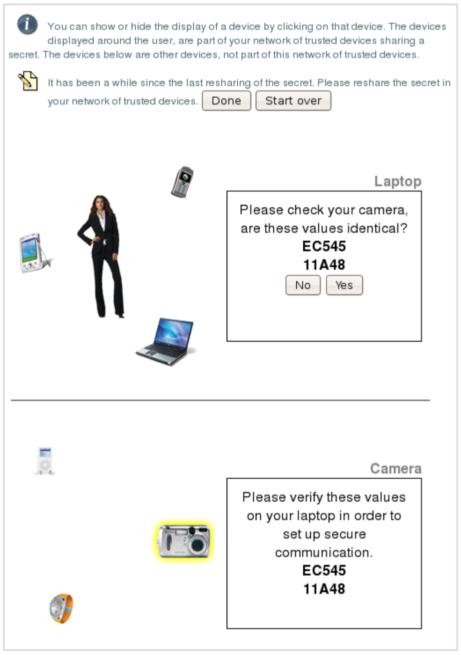
1. "It has been a while since the last resharing of the secret. Please reshare the secret in your network of trusted devices".
2. "You want to sell your laptop, but first remove it from your network of trusted devices".
3. "You bought an mp3-player and would like to add it to your network of trusted devices".

In order to evaluate the user interface we recruited two experts in the field of *Human Computer Interface* (HCI) to do an expert evaluation. Before doing user observation the designed interface was altered according to suggestions in the expert review.

During the preliminary test, test subjects were asked to think aloud in order to identify possible problems with the user interface. The satisfaction of the usability was measured by the *After Scenario Questionnaire* (ASQ), based on the work of Lewis [108]. While the user completed the different scenarios, additional logging was executed to measure other usability factors such as effectiveness and efficiency [89].

Web-based interface

Figure 4.7 depicts the web-based interface for our user experiments. The devices around the user are part of the threshold secret sharing. The devices below represent devices in the neighbourhood, not part of the threshold secret sharing.


You can show or hide the display of a device by clicking on that device. The devices displayed around the user, are part of your network of trusted devices sharing a secret. The devices below are other devices, not part of this network of trusted devices.

It has been a while since the last resharing of the secret. Please reshare the secret in your network of trusted devices.

Please check your camera, are these values identical?
EC545
11A48

Please verify these values on your laptop in order to set up secure communication.
EC545
11A48

Figure 4.7: Web-based Interface.

To simulate the real world situation where the user needs to get his mobile phone out of his pocket before he can see its display, he is required to click on a device before its display is shown. Additionally this allows us to know with which devices' interfaces the test subject is interacting.

Mock-up interface

The protocol needs to be extended with a graphical user interface for the end-user to know what is going on. We implemented a minimalistic interface where we assumed that each device display is identical and only has a relatively low resolution. The starting screen for each device is given in Fig. 4.8.

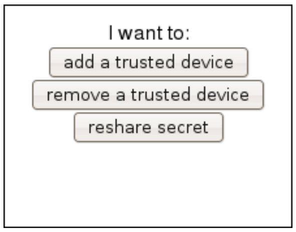


Figure 4.8: Starting Display for Devices Part of the Threshold Secret Sharing.

As soon as the end-user selects one of the options on any device, this device will act as the proxy and the request will displayed on all other devices in a user-friendly way (step 2a of the protocol), as depicted in Fig. 4.9. This displayed request allows the user to visually confirm his request. To avoid abuse, it contains the proxy and the request. Buttons to confirm or deny the request are provided.

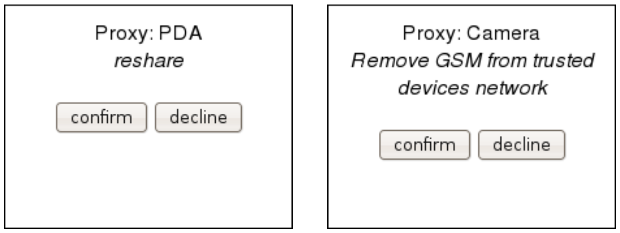
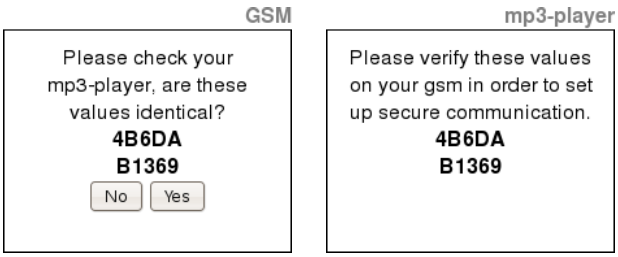


Figure 4.9: Request Displayed in User Friendly Way.

When a new device, e.g., an mp3-player is added a MANA protocol needs to be carried out. Figure 4.10 displays our design.

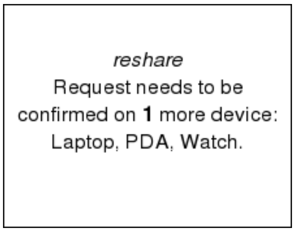


Please check your mp3-player, are these values identical?
4B6DA
B1369

Please verify these values on your gsm in order to set up secure communication.
4B6DA
B1369

Figure 4.10: MANA Protocol to Add a New Device.

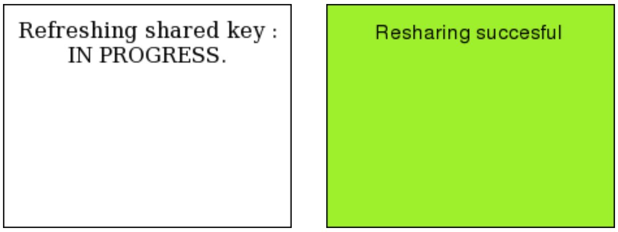
After confirming, if the request needs to be confirmed on more devices, supporting information is displayed (see Fig. 4.11).



reshare
Request needs to be confirmed on **1** more device:
Laptop, PDA, Watch.

Figure 4.11: Example of Support Information.

After the user confirmed his request at t devices (step 2c of the protocol), the actual resharing protocol is executed and the following screens (see Fig. 4.12) are displayed. Each screen is displayed for 3 seconds. The first delay simulates the time it takes to do all calculations and communication necessary to complete the protocol. The second delay allows the end-user to see that his request was processed successfully.



Refreshing shared key : IN PROGRESS.

Resharing succesful

Figure 4.12: Screens at the End of the Protocol.

No screens to indicate failure of the resharing protocol were included in this preliminary study. None of the devices behaved maliciously, as this might further confuse the test subjects. However, the way the end-user handles this malicious behaviour is very important to the overall security. Future tests would benefit from having test subjects returning at a later point in time, when they are already familiar with the system, to do a similar test that includes devices behaving maliciously with a certain probability.

Expert review

Before conducting any user observations we subjected the system interface to an expert review to expose the most obvious usability issues in an early design phase. The two HCI experts have no specific knowledge of the domain of security systems. In this case, the lack of domain experience is not considered to be a drawback since managing or configuring security systems is usually not a primary goal for most people [171]; it is only secondary to achieving real goals such as accessing some documents. The expert study took place in the reviewers' own office environment, on an individual basis after which results were compared.

As a guideline, the expert reviewers followed Nielsen's ten usability heuristics [117] and Yee's key principles for secure interaction design [174]. Although the interface is somewhat limited in the amount of functionalities offered, the review indicated some potential usability obstacles:

- Match between system and the real world [117]. A significant usability problem was located at the very first point-of-contact between the system and the user. People with a security background might not be the typical end users of the proposed security scheme. In this light, it is undesirable to confront your end users with any technical details about the algorithms behind the security system. To explain the scheme, the reviewers rewrote system-oriented terms, e.g., "threshold secret sharing" to match more familiar concepts, e.g., "network of trusted devices".
- Visibility of system status [117]. When a user is performing a MANA protocol, it is often necessary to switch between mobile devices to successfully complete the process. However, this was not clearly represented in the GUI. Now, whenever the user needs to switch devices, the revised system will list all appropriate devices as well as light up their graphical representations in the GUI.
- User control and freedom [117]. Some MANA protocols can be rather tedious, which may cause the user to feel lost at certain points. As a consequence undo-functions going one step back in the process are required, next to the already provided "reset" functionality.

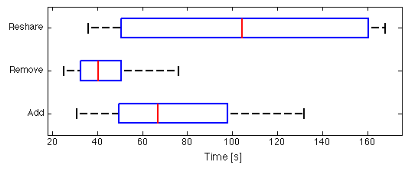
- Explicit authorisation [174]. Users granting or removing authorisations to/from other actors must unambiguously know the consequences of their actions. On that account, many labels (buttons, titles, etc.) have been revised, e.g., when adding a new device to the threshold the button “next” has become “add” to prevent users from assuming there will be another step in a wizard-like setting.

Other heuristics were applied but are not mentioned here as they only resulted in some minor additional changes.

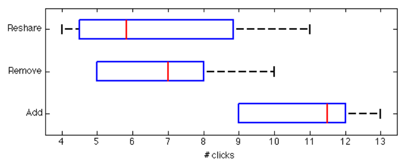
Preliminary study

The preliminary study was conducted among eight students[4] from different backgrounds (technical as well as non-technical). The test subjects were recruited on a voluntary basis in public study-oriented areas of different faculties. The motivation behind the choice for a student population, was their openness to new concepts. Especially since the concept of threshold personal devices needed to be explained first, before being able to conduct this study. This study was too limited to draw conclusions, but gave some good indications about what needed to be changed and directions for further studies.

Data from the additional logging. Figure 4.13 gives an overview in the form of a box-plots of the time and number of clicks it took the test subjects to complete the three scenarios. The minimum number of clicks required to complete each scenario successfully are: four for resharing the secret, five for removing a device and seven for adding a device. None of the user’s succeeded in adding a device with the minimum number of clicks, this points to a discrepancy between our designed interface and the user’s mental model.



(a) Time.



(b) Number of clicks.

Figure 4.13: Additional Information from Logging.

[4]A special thanks to Alexander, Charissa, Elke, Koen, Iris, Johan, Lotte and Pieter for their participation.

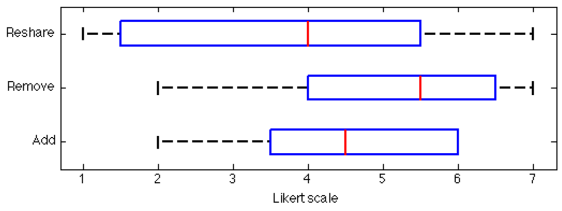
Data from the ASQ. The After Scenario Questionnaire (ASQ) consists of three statements:

- Overall, I am satisfied with the ease of completing the tasks in this scenario;
- Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario;
- Overall, I am satisfied with the support information (messages) when completing the tasks.

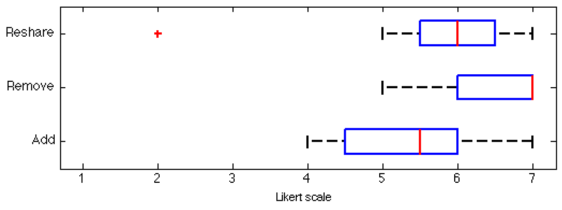
The test subjects responded to these statements on a seven point Likert scale, from strongly disagree (1) to strongly agree (7). These statements are presented to the test subject right after completing each individual scenario. The results are shown in Fig. 4.14.

Findings. The three scenarios are ranked in the order these were presented to the test subject. Data clearly show the confusion among the test subjects for the first scenario. Note that this was also the first time they were confronted with the web-based interface. The small informative text on how to interact with the devices was not sufficient for most test subjects to get started. Future tests would benefit of first having the test subjects to get familiar with the interface, e.g., by showing typical device-specific screens when opening displays. This would allows to get rid of the bias.

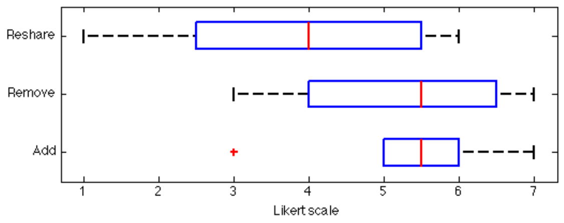
Scenario 1: Reshare. All test subjects were confused and did not know what to do. This can only partly be attributed to the fact that it was the first time that they interacted the web-based interface. However, the test subjects also had grave difficulties with the resharing scenario. These difficulties came somewhat as a surprise, since resharing is the most simple of the three scenarios from the security point of view. From a usability point of view, this scenario has the smallest number of required interactions with the user. However, it is hard for the user to picture resharing. Most test subjects indicated not to know what they just did. A typical user will not authorise a resharing because this can be authorised automatically. The option to authorise resharing is included for more advanced users. The devices GUI reflects this, see Fig. 4.8. As this is intended for more advanced users, this scenario should have been the last scenario. The goal of resharing (without adding or removing a device), to renew the shared key, can be made more explicit by abstracting away from resharing and introducing the "refreshing" of a shared key.



(a) . . . the ease of completing the tasks in this scenario".



(b) . . . the amount of time it took to complete the tasks in this scenario".



(c) . . . the support information (messages) when completing the tasks".

Figure 4.14: Additional Information from ASQ. Test subjects responded to the following statements: “Overall, I am satisfied with . . .

Scenario 2: Remove a device. Removing a device was generally considered an easy and straightforward scenario.

When removing a specific device, more or less half of the test subjects tend to use this device as the proxy. In general, other scenarios exist in which one should also remove a device from the set of trusted devices, e.g., lost or broken devices. In these cases the device to be removed cannot be used as a proxy. We did not allow for the proxy to remove itself because it did not seem sensible to help authorise a resharing where this device will not be part of. However, when selling a device, one wants to make sure that this device is indeed removed from the set of trusted devices. Another possible explanation was given by one of the test subjects: “When you want to disconnect your external hard drive from your computer, you first need to safely remove it by clicking on the icon representing that hard drive". For future testing, the interface is updated to allow for the device that is to be removed to act as a proxy.

Scenario 3: Add a device. Adding a device proved more difficult. The actions for adding a device used to consist of: a manual authentication step between the new device and one of the personal devices; a confirmation step on the threshold number of personal devices; and finally a verification step on the threshold number of personal devices. This clearly put quite a high burden on the user. We redesigned the protocol for authorisation to make use of a group authentication protocol. Therefore, we got rid of the verification step.

The user could only start adding a device from a personal device, but all test subjects wanted to be able to start from the device to be added.

We also learnt that the values for a user to compare in the manual authentication step should not be displayed in two groups on one display, e.g. in two consecutive lines (see Fig. 4.10). Some thought that they needed to compare these two values instead of comparing the values across displays. Most of the confused test subjects did not look at the display of the new device at the time of confusion. One possible solution is to put both check values as one string, possibly using an alternative encoding to keep the string short.

General remarks. The variation in the number of clicks can be partially attributed to test subjects who wanted to open or close more displays.

Several test subjects indicated that the support information in the form of messages really helped them by pointing out the right devices. One subject was a bit confused by the message displayed in Fig. 4.11. He was not sure whether he was supposed to confirm his request on one device or on all suggested devices.

After completion of the manual authorisation for removing this device, the GUI of participating devices displays "Resharing in progress" and "Resharing successful". One test subject asked why the GUI indicated resharing when he had removed a device. Although only one remarked this, later questions on the mental model revealed that the majority of the test subjects think of simply resharing the key, adding devices and removing devices as three different operations. This also clarified that there should be a clear distinction between the underlying protocol, resharing, and the provided options that caused the resharing. The interface is updated accordingly, now showing "refreshing shared key", "removing X from set of trusted devices" or "adding X to set of trusted devices".

Most test subjects did not see why one should add an mp3-player. They deemed Threshold Things That Think useful as a means of protecting data, but they would only deploy it with the devices that contain sensitive data, like for instance a mobile phone, PDA or laptop. One question of one of the test subjects “What if one of the devices breaks down?” is another example showing that the user does not grasp the underlying mechanism, threshold cryptography.

Further usability studies would benefit from a broader context, e.g., just introducing the concept of threshold personal devices that can be used to log on to a website and comparing this with traditional means of logging in. With RFID-stickers, test subjects can transform any personal object into a smart object.

4.5 Conclusion

In this chapter, we have stressed the need for resharing protocols for things that think. These protocols do not only allow for periodic updates all devices’ shares, hence limiting the time frame for attackers. Moreover, resharing protocols also enable one to add or remove devices from the set of personal devices. We showed how to reshare a secret-shared secret with devices that are absent at the moment of resharing. Furthermore, we proposed a protocol that allows the end-user to authorise resharing and hence be really in control of his overall security system. We designed a secure and user-friendly protocol. We took the opportunity to create one possible interface for simulating the protocol and conducted a preliminary usability study on it.

Chapter 5

Private RF IDentification

To ensure that the end-user cannot be tracked easily by all the communication between his personal devices and the environment, enabling private authentication over an open channel is important. More specifically, we look into this problem from the point of view of the least powerful devices, namely RFID tags. This chapter introduces a new RFID model that supports multiple readers and mutual authentication. We also propose an efficient wide-strong RFID identification protocol that is based on zero-knowledge. Subsequently we modify this protocol such that it also achieves efficient mutual authentication.

5.1 Introduction

Radio Frequency Identification (RFID) systems are becoming more common. RFID tags are deployed in various consumer applications such as physical access tokens, car keys, contactless payment systems and electronic passports. For these applications, it is crucial that the underlying protocols protect not only security but also protect the (location) privacy of the end user (see Weis *et al.* [166]). Yet, all communication with RFID tags can easily be eavesdropped or modified, tags respond to any query and RFID tags can be corrupted, which renders these vulnerable to attacks. On top of this, an adversary can typically learn the outcome of the identification protocol. Successful identifications result in an unlocked door, unlocked car or processed payment, while failure has no outcome. Since RFID tags are primarily used for authentication purposes, security in this context means that it should be infeasible to impersonate a legitimate tag.

Privacy, on the other hand, means that unauthorised parties should not be able to identify, trace, or link tag appearances.

Several models for privacy and security in the context of RFID systems have been proposed in the literature. We only consider the *general* models. For some of these models it is shown that, despite their intended generality, it remains unclear how to apply them to protocols other than the protocol in the context of which they were proposed. Other existing models do not allow for adversaries that can tamper with tags. However, considering such adversaries is important because, as low-cost devices, tags are hardly protected against physical tampering. In particular, it has been shown that side-channel attacks may enable an adversary to extract secrets from the tag [88, 99, 134, 113], and so-called 'reset' attacks force the tag to re-use old randomness [17, 37, 76]. The adversary can mount reset attacks by inducing power drops or by otherwise influencing the physical environment of the tag. Adversaries that can tamper with tags are therefore realistic.

So far, little attention has been paid to supporting multiple readers. Most RFID models and protocols only take into account the limited setting of one reader. Multiple readers occur for example when using a single RFID card to access multiple disjoint security systems (e.g., multiple buildings, printer systems, vending machines). Supporting multiple readers, however severely complicates the setup since it is more likely that one of the readers will be compromised. This should not affect the security and passive privacy of other readers and the tags. Hence, sharing secrets among readers is impossible. In this setting mutual authentication between tag and reader is a desirable property. Moreover, supporting multiple readers is a stepping stone to providing device-to-device authentication.

Our new model is an extension of the model of Hermans *et al.* [84], to support multiple readers and mutual authentication. We believe that this extension preserves the benefits of the original: general, robust and easily applicable. Using this new model as a reference we design and evaluate a new wide-strong private RFID identification protocol that supports mutual authentication and can handle multiple readers.

We design and evaluate an efficient RFID identification protocol with the strongest possible privacy guarantees, i.e. wide-strong. The notion of strong privacy means that no adversary actively interacting with the tags and the reader, is able to infer any information on a tag's identity from tag communication, even when given all secrets stored on the tag. The notion of wide-strong privacy corresponds to strong privacy against adversaries that also learn the outcome of the protocol. This privacy notion cannot be achieved when considering only symmetric identification protocols, where some cryptographic secret is shared between tag and reader. Examples of such protocols can be found in [20, 72, 94]. The main reason behind using symmetric identification protocols is the perception that public-key cryptography requires either too much time, power or circuit area to implement on low-cost devices. However, Lee *et al.* [106] and Hein *et al.* [82] showed that public key cryptography, in particular Elliptic Curve Cryptography (ECC), can be realised on RFID tags. Previously, wide-strong privacy has only been achieved by schemes relying on an IND-CCA2 encryption scheme (or variants of such schemes) [33, 162]. Our scheme only needs an ECC architecture without additional components typically required for IND-CCA2 encryption (e.g. hash function), resulting in a smaller hardware footprint, which is a substantial improvement. We also propose a modification of this protocol that provides mutual authentication.

5.2 Previously Proposed Models

This section discusses certain existing RFID privacy models. These models usually consist of a correctness (no false negatives), security (no false positives) and privacy definition.

We use a common model for RFID systems, similar to the definitions introduced by Vaudenay [162] and Canard *et al.* [33]. For our model, we extend these definitions to allow for multiple readers, unlike previously proposed models. An RFID system consists of a set of tags $\mathcal{T}$ and a set of readers $\mathcal{R}$. Each tag is identified by an identifier ID. The memory of the tags contains a state S, which may change during the lifetime of the tag. The tag's ID may or may not be stored in S. Each tag is a transponder with limited memory and computation capability.

Tags can also be corrupted: the adversary has the capability to extract secrets and other parts of the internal state from the tags it chooses. A reader R_i consists of one or more transceivers and a database. A reader's task is to identify legitimate tags (i.e. to recover their IDs), and to reject all other incoming communication. A tag is 'registered' with a reader if the reader database contains an entry for that tag and can successfully authenticate it.

An RFID system requires several algorithms and protocols for setting up the readers, tags, registering tags with readers or even unregistering tags. These routines are highly dependent on the specific architecture of the RFID system and can even take place offline. The privacy and security of these setup and registration algorithms and protocols is outside of the scope of this thesis. Therefore, we will only discuss the main protocol used for tag identification and none of the auxiliary setup and registration routines.

Many models, including the ones introduced in [10, 31, 46, 78, 95, 161] do not allow corrupted tags to be traced. We have selected one such model [95] for further discussion, in addition to the stronger models of Vaudenay [162] and Canard et al. [33].

5.2.1 Vaudenay

Several concepts from the privacy model introduced by Vaudenay [162] are used in our model. We therefore present Vaudenay's model in detail.

Adversarial model

The adversary of the Vaudenay model has the ability to influence all communication between a tag and the reader and can therefore perform man-in-the-middle attacks on any tag that is within its range. It may also obtain the result of the authentication of a tag, i.e. whether the reader accepts or rejects the tag. The adversary may also "draw" (at random) tags and then "free" them again, moving them inside and outside its range. During these interactions the adversary has to use a virtual identifier (not the tag's real ID) in order to refer to the tags that are inside its range. Finally the adversary may corrupt tags, thereby learning their entire internal state.

The above interactions take place over eight oracles that the adversary may invoke: `CreateTag(ID)`, `DrawTag(distr)` $\rightarrow$ $(vtag)$, `Free(`$vtag$`)`, `Launch` $\rightarrow \pi$, `SendReader(`m, π`)` $\rightarrow m'$, `SendTag(`$m, vtag$`)` $\rightarrow m'$, `Result(`π`)` $\rightarrow x$ and `Corrupt(`$vtag$`)`. $vtag$ denotes a virtual tag reference, π a protocol instance, *distr* a polynomially bounded sampling algorithm, m and m' messages and ID a tag ID. For a complete definition of the oracles the reader is referred to [162].

The Vaudenay model divides adversaries into different classes, depending on restrictions regarding their use of the above the oracles. In particular, a *strong* adversary may use all eight oracles without any restrictions. A *destructive* adversary is not allowed to use a tag after it has been corrupted. This models situations where corrupting a tag leads to the destruction of the tag. A *forward* adversary can only do other corruptions after the first corruption. That is, no protocol interactions are allowed after the first corrupt. A *weak* adversary does not have the ability to corrupt tags. Orthogonal to these four attacker classes there is the notion of *wide* and *narrow* adversary. A *wide* adversary has access to the result of the verification by the server while a *narrow* adversary does not.

Due to their generality, the above restrictions can be used perfectly in other privacy models. Throughout this chapter we will frequently refer to strong, destructive, forward, weak and wide/narrow adversaries.

The equations below show the most important relations between the above privacy notions:

$$\begin{tikzcd}
\text{wide-strong} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{wide-destructive} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{wide-forward} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{wide-weak} \arrow[d, Rightarrow] \\
\text{narrow-strong} \arrow[r, Rightarrow] & \text{narrow-destructive} \arrow[r, Rightarrow] & \text{narrow-forward} \arrow[r, Rightarrow] & \text{narrow-weak}\,.
\end{tikzcd}$$

In this case $A \Rightarrow B$ means that if the protocol is A-private it implies that the protocol is B-private. A protocol that is *wide-strong* private, for example, obviously also belongs to all other privacy classes, that only allow weaker adversaries.

Privacy, security and correctness

In general, an RFID protocol should satisfy (a) correctness (a real tag is always accepted), (b) security (fake tags are rejected) and (c) privacy (tags cannot be identified or traced). Privacy is defined by means of the notion of a 'trivial' adversary.

Definition 4 (Blinder, trivial adversary - Simplified version of Definition 7 from [162])**.** *A Blinder $\mathcal{B}$ for an adversary $\mathcal{A}$ is a polynomial-time algorithm which sees the messages that $\mathcal{A}$ sends and receives, and simulates the* `Launch`*,* `SendReader`*,* `SendTag` *and* `Result` *oracles to $\mathcal{A}$. The blinder does not have access to the reader tapes. A blinded adversary $\mathcal{A}^{\mathcal{B}}$ is an adversary that does not use the* `Launch`*,* `SendReader`*,* `SendTag` *and* `Result` *oracles.*

An adversary $\mathcal{A}$ is trivial if there exists a blinder $\mathcal{B}$ such that $|\Pr(\mathcal{A}\, wins) - \Pr(\mathcal{A}^{\mathcal{B}}\, wins)|$ is negligible.

Intuitively, an adversary is called trivial if, even when blinded, it still produces the same output. Such an adversary does not 'use' the communication captured during the protocol run in order to determine its output. Note that a blinded adversary is not the same as a simulator typically found in security proofs: the blinder is independent of the adversary and has no access to the adversary's tape. The blinder just receives incoming queries from the adversary and has to respond either by itself or by forwarding the queries to the system.

We are now ready to present the privacy definition.

Definition 5 (Privacy - Simplified version of Definition 6 from [162])**.** *The privacy game between the challenger and the adversary consists of two phases:*

1. *Attack phase: the adversary issues oracle queries according to applicable restrictions;*
2. *Analysis phase: the adversary receives the table that maps every vtag to a real tag ID. Then it outputs* `true` *or* `false`*.*

The adversary wins if it outputs `true`*. A protocol is called P-private, where P is an adversary class (strong, destructive, ...), if and only if all winning adversaries belong to the class P are trivial.*

Besides privacy the protocol should also offer authentication of the tag. We refer to this property as the *security* of the protocol.

Definition 6 (Security - Simplified version of Definition 4 from [162])**.** *We consider any adversary in the class* strong*. The adversary wins if the reader identifies an uncorrupted legitimate tag, but the tag and the reader did not have a matching conversation. The RFID scheme is called secure if the success probability of any such adversary is negligible.*

Definition 7 (Correctness - Definition 1 from [162])**.** *An RFID scheme is correct if its output is correct except with negligible probability for any polynomial-time experiment which can be described as follows:*

1. *Set up the reader;*
2. *Create a number of tags including a subject one named ID;*
3. *Execute a complete protocol between reader and tag ID.*

The output is correct if and only if Output $=\perp$ and tag ID is not legitimate or Output $= ID$ and tag ID is legitimate.

Discussion

The paper of Vaudenay inspired many authors to formulate derived RFID privacy models or to evaluate the Vaudenay model [28, 33, 47, 48, 119, 120, 124, 139, 138]. Although Vaudenay's privacy model is perhaps the strongest and most complete, it contains some flaws with respect to strong privacy.

Vaudenay's proof of the statement that 'strong privacy is impossible' uncovers some of these shortcomings. This proof assumes a destructive private protocol. By definition, for every destructive adversary, there exists a blinder. This includes the adversary that (a) creates one real tag, (b) corrupts this tag right away, (c) starts a protocol using either the state from the corrupted tag or from another fake tag. In the end, the blinder has to answer the `Result` oracle. Obviously, the adversary knows which tag was selected and knows which result to expect. However, since the blinder has no access to this random coin of the adversary, it must be able to distinguish a real and a fake tag just by looking at the protocol run from the side of the reader. The proof then uses this blinder to construct a strong adversary. Since all strong adversaries are also destructive, this proves the impossibility of strong privacy.

Obviously, this proof only works because the blinder is separated from the adversary. In later work [163], Vaudenay corrects the inconsistency in the model and shows that strong privacy is indeed possible. In this new approach, the blinder is given access to the random coin flips of the adversary. The issue with a separate blinder is exploited multiple times by Armknecht et al. in [6]. Using this property the authors show the impossibility of reader authentication combined with respectively narrow forward privacy (if `Corrupt` reveals the temporary state of tags) and narrow strong privacy (if `Corrupt` only reveals the permanent state of tags).

Independent from this correction, Ng et al. [119] also identified the problems with strong privacy. They propose a solution, based on the concept of a 'wise' adversary that does not make any 'irrelevant' queries to the oracles i.e. queries to which it already knows the answer. The authors claim that, if the protocol does not generate false negatives, then a wise adversary never calls the `Result` oracle. Given the vague definition of wise adversaries it is hard to verify these claims. The existence of attacks which exploit false positives [22] however, suggests that the general claim that `Result` is not used by a wise adversary is incorrect. Based on this questionable general claim, the authors further identify an IND-CPA-based protocol as being strong private, without giving a formal proof[1].

[1]Note that the original security proof (i.e. no false positives) by Vaudenay requires IND-CCA2 encryption, so using only IND-CPA encryption would require a new security proof. The `Result` may therefore serve as a decryption oracle.

5.2.2 Canard et al.

Model

The model of Canard et al. [33] builds on the work of Vaudenay, so the definition of oracles is quite similar. For the privacy definition the model requires the adversary to produce a non-obvious link between virtual tags.

Definition 8. $(vtag_i, vtag_j)$ *is a non-obvious link if* $vtag_i$ *and* $vtag_j$ *refer to the same ID and if a 'dummy' adversary, who only has access to* `CreateTag`*,* `Draw`*,* `Free`*,* `Corrupt`*, is not able to output this link with a probability better than* $1/2$[2]*.*

One major difference with respect to Vaudenay's model is that a 'dummy' adversary is used instead of a blinded adversary. This avoids some of the issues surrounding the use of a blinder, because a 'dummy' adversary can also access its own random tape, while a blinder cannot access the adversary's random tape.

The definition requires the adversary to output a non-obvious link. A protocol is said to be untraceable if, for every adversary $\mathcal{A}$, it is possible to construct a 'dummy' adversary $\mathcal{A}_d$ such that $|\mathbf{Succ}_{\mathcal{A}}^{Unt}(1^k) - \mathbf{Succ}_{\mathcal{A}_d}^{Unt}(1^k)| \leq \epsilon(k)$.

Discussion

While the work certainly has its merit in formalizing and fixing the Vaudenay model (by using a dummy adversary instead of a blinder), the model of Canard et al. lacks generality because it focuses on non-trivial links. Other relevant properties, which do not imply the leakage of a non-trivial link, are not considered a privacy breach. For example, the cardinality of the set of active tags can be leaked without leaking a non-trivial link. Because of the limited scope of untraceability, we are not using this model.

[2] It is unclear why the authors use the probability threshold $1/2$, since one would expect some dependency on the total number of non-obvious links. One slightly different interpretation is that a 'dummy' adversary cannot determine if a given non-obvious candidate link $vtag_i$, $vtag_j$ is a link in reality or not.

5.2.3 Juels-Weis

Model

The Juels-Weis model [95] (see also [96]) is based on the notion of indistinguishability. The model does not feature a `DrawTag` query and the `Corrupt` query is replaced by a `SetKey` query, which returns the current secret of the tag and allows the adversary to set a new secret. Figure 5.1 shows a simplified version of the privacy game. The protocol is considered private if $\forall \mathcal{A}$, $\Pr\left[\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}} \text{guesses } b \text{ correctly}\right] \leq \frac{1}{2} + \epsilon$.

Experiment $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}$:

1. Setup:
   - Generate n random keys key_i.
   - Initialise the reader with the random key_i.
   - Create n tags, each with a key_i.
2. Phase (1): Learning
   - $\mathcal{A}$ can interact with a polynomial number of calls to the system, but can only issue `SetKey` on $n-2$ tags, leaving at least 2 uncorrupted tags.
3. Phase (2): Challenge
   - $\mathcal{A}$ selects two uncorrupted tags T_0 and T_1. Both are removed from the set of tags.
   - One of these tags (T_b, the challenge tag) will be selected at random by the challenger.
   - $\mathcal{A}$ can make a polynomial number of calls to the system, but cannot corrupt the challenge tag T_b.
   - $\mathcal{A}$ outputs a guess bit $g \in \{0,1\}$.

Figure 5.1: Privacy Experiment from Juels-Weis[95].

Discussion

The Juels-Weis model is one of the few models based on a simple indistinguishability game instead of the notion of simulatability. The model is limited by the fact that the challenge tags cannot be corrupted. In terms of the model of Vaudenay [162], it would be a Weak adversary with regard to the challenge tags. However, the Juels-Weis model is slightly stronger with regard to insider attackers, and corresponds to Weak-Insider as defined in the new model (see next Sect. 5.3). For example, attacks in which the adversary links together executions of a tag that have taken place prior to its corruption are not possible in the Juels-Weis model, but the adversary can use a corrupted tag to link other uncorrupted tags.

The model from [78] is very similar, with the difference that the privacy is defined as distinguishing the reply from a real tag from a random reply.

5.2.4 Bohli-Pashalidis

Model

Unlike the previous models, the Bohli-Pashalidis model [25] is not an RFID-specific model. Unfortunately, it captures only privacy properties; properties such as security and correctness are not covered. The model considers a set of users (with unique identifiers) $\mathcal{U}$, whose size is at least polynomial in a security parameter. There is no formal difference between different types of players, like there is with tag and reader in most RFID models. The system $\mathcal{S}$ can be invoked with input batches $(u_1, \alpha_1), (u_2, \alpha_2), \ldots, (u_c, \alpha_c) \in (\mathcal{U}, A)^c$, consisting of pairs of user identifiers and 'parameters' and will output a batch $((e_1, \ldots e_c), \beta)$, with the outputs e_i from each system invocation and a general output β, applying to the batch as a whole. Users can also be corrupted, revealing their internal state to the adversary.

The authors investigate the properties of the function $f \in \mathcal{F}$, where $\mathcal{F} = \{f : \{1, 2, \ldots, n\} \rightarrow \mathcal{U}\}$ is the space of functions that map the serial number of each output element to the user it corresponds to. In the Strong Anonymity (SA) setting, no information should be revealed to the adversary about the function f, guaranteeing the highest level of privacy. Several weaker notions (which reveal *some* information on f) are defined and the relations among notions are examined.

In the RFID setting the batch properties are currently not considered, although this would be an interesting extension, since some localization protocols are based on batch invocations of a large set of RFID tags. For simplicity we restrict ourselves to the Bohli-Pashalidis model for online systems. For these systems, where all batches have size one (i.e. the system never waits for multiple inputs until it produces some output), the only two applicable distinct notions are Strong Anonymity (SA) and Pseudonymity (PS).

The adversarial model is based on indistinguishability. The adversary can cause different users to invoke the system using different parameters (e.g. messages) in both a left and right world with the `Input`$((u_0, \alpha_0), (u_1, \alpha_1))$ oracle. Based on a bit b, selected by the challenger, the system will be invoked with the user-data pair (u_b, α_b). That is, the adversary itself defines the functions $f_0, f_1 \in \mathcal{F}$, for respectively the left and right world. The adversary can also corrupt users. At the end of the game the adversary has to output a guess bit g. The adversary wins the game if $g = b$. By imposing restrictions on f_0 and f_1, the authors investigate different levels of privacy.

Definition 9. *A privacy protecting system $\mathcal{S}$ is said to unconditionally provide privacy notion X, if and only if the adversary $\mathcal{A}$ is restricted to invocations (u_0, α_0) and (u_1, α_1) such that f_0 and f_1 are X-indistinguishable for all invocations and for all such adversaries $\mathcal{A}$, it holds that $\mathbf{Adv}^X_{\mathcal{S},\mathcal{A}}(k) = 0$.*

Similar definitions for computational ($\mathcal{A}$ is polytime in k and $\mathbf{Adv}^X_{\mathcal{S},\mathcal{A}}(k) \leq \epsilon(k)$) and statistical privacy are available.

Discussion

As it is general and not meant to cover security properties, the Bohli-Pashalidis model needs non-trivial adaptations in order to apply to the RFID setting. In its current form, the model does not support multi-pass protocols, where linking two messages from the same protocol run is not a privacy breach. Moreover there is no distinction between tags that need to be protected, and the reader for which privacy is not an issue. An interesting question is whether the strictly binary distinguishing game (only one bit of randomness in the challenge) provides enough flexibility compared to other models, like Vaudenay's, where there are multiple bits of randomness that are to be guessed.

5.3 A New Model

Our proposed model is an extension of the privacy model of Hermans *et al.* [84], hence this model is not discussed separately. This model was selected since it is based on the well-studied notion of (left-or-right) indistinguishability. This avoids the issues with less well-studied concepts such as blinders that the Vaudenay model suffers from (see Sect. 5.2.1). Moreover, since several cryptographic schemes have proven security properties based on indistinguishability games (e.g., IND-CPA, IND-CCA, IND-CCA2), this is likely to simplify the proofs when using these schemes as building blocks. Note that the Juels-Weis model from Sect. 5.2.3 also uses a traditional indistinguishability setup. However, the model requires the adversary to distinguish one out of two selected tags in the final phase. The disadvantage of this approach is that it does not take into account other properties that might leak privacy (e.g. cardinality) and that it limits the use of tag corruption. The Vaudenay model did introduce some crucial tools such as virtual tag references and the corruption types that are still required.

This extended model allows for a more general setting where a tag can privately authenticate to multiple (independent) readers. By incorporating the creation of insider tags (valid tags that are under the control of an adversary), our extended model also captures insider attacks, as introduced by Van Deursen *et al.* [160]. For protocols that are vulnerable to these kind of attacks, the adversary can link other legitimate tags, using its insider tag and the `Result` oracle.

We assume a set of readers $\mathcal{R} = \{R_1, R_2, \ldots, R_j\}$ and a set of tags $\mathcal{T} = \{T_1, T_2, \ldots, T_i\}$. $\mathcal{R}$ and $\mathcal{T}$ are initially empty, and readers and tags are added dynamically by the adversary. Each reader maintains a database of tuples (ID_i, K_i), one for every tag $T_i \in \mathcal{T}$ that is registered with that reader. Moreover, every tag T_i stores an internal state S_i.

Let $\mathcal{A}$ denote the adversary, which can adaptively control the system $\mathcal{S}$. The adversary interacts with the system through a set of oracles:

- `CreateReader()` $\to R_j$: this oracle creates a new reader. A reference R_j to the new reader is returned.
- `CreateTag(ID)` $\to T_i$: on input a tag identifier ID, this oracle creates a tag with the given identifier and corresponding secrets. A reference T_i to the new tag is returned. Note that this does not reject duplicate IDs.
- `RegisterTag(`T_i`,`R_j`)`: register the tag T_i with the server R_j. The registration of the tag with the reader can be done in several ways (e.g. using a specific protocol that involves both the tag and reader, between readers or using some offline process).
- `Launch(`R_j`)` $\to \pi$: this oracle launches a new protocol run on the reader R_j, according to the protocol specification. It returns a session identifier π, generated by the reader.

- `DrawTag(`T_i`,`T_j`)` $\rightarrow$ *vtag*: on input a pair of tag references, this oracle generates a virtual tag reference, as a monotonic counter, *vtag* and stores the triple $(vtag, T_i, T_j)$ in a table $\mathcal{D}$. Depending on the value of b, *vtag* either refers to T_i or T_j. If one of the two tags T_i or T_j is in the table $\mathcal{I}$, $\perp$ is returned and no entry is added to $\mathcal{D}$. If T_i is registered with a different set of readers than T_j, $\perp$ is returned. If T_i is already references as the left-side tag in $\mathcal{D}$ or T_j as the right-side tag, then this oracle also returns $\perp$ and adds no entry to $\mathcal{D}$. Otherwise, it returns *vtag*.

- `Free(vtag)`$_b$: on input *vtag*, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$. If $b = 0$, it resets the tag T_i. Otherwise, it resets the tag T_j. Then it removes the entry $(vtag, T_i, T_j)$ from $\mathcal{D}$. When a tag is reset, its volatile memory is erased. The non-volatile memory, which contains the state S, is preserved.

- `SendTag(vtag,`m`)`$_b$ $\rightarrow m'$: on input *vtag*, this oracle retrieves the triple $(vtag, T_i, T_j)$ from the table $\mathcal{D}$ and sends the message m to either T_i (if $b = 0$) or T_j (if $b = 1$). It returns the reply from the tag (m'). If the above triple is not found in $\mathcal{D}$, it returns $\perp$.

- `SendReader(`R_j`, `π`, `m`)` $\rightarrow m'$: on input π, m this oracle sends the message m to the reader R_j in session π and returns the reply m' from the reader (if any) is returned by the oracle[3].

- `Result(`R_j`,`π`)`: on input π, this oracle returns a bit indicating whether or not the reader accepted session π as a protocol run that resulted in successful authentication of a tag. If the session with identifier π is not finished yet, or there exists no session with identifier π, $\perp$ is returned.

- `Corrupt(`T_i`)`: on input a tag reference T_i, this oracle returns the complete internal state of T_i[4]. Note that the adversary is not given control over T_i.

- `CreateInsider(ID)` $\rightarrow T_i, S$: create an insider tag T_i. This runs `CreateTag` to create a new tag T_i and `Corrupt` on the newly created tag. The tag T_i is added to the list $\mathcal{I}$ of insider tags.

- `CorruptReader(`R_j`)`: corrupt the reader R_j, which returns the full database of the reader and all secrets of the reader. Note that in the default privacy game this oracle is not used.

[3]If no active session π exists, the reader is likely to return $\perp$.

[4]Both the volatile and non-volatile state is returned. For multi-pass protocols it might be necessary to relax this to only the non-volatile state; to force the adversary to only corrupt tags T_i that are currently not drawn; or to use the concept of X^+ privacy, as discussed in Sect. 5.3.3.

The experiment that the challenger sets up for $\mathcal{A}$ (after the security parameter k is fixed) proceeds as follows:

$\mathbf{Exp}_{\mathcal{S},\mathcal{A}}(k)$:

1. $b \in_R \{0,1\}$
2. $g \leftarrow \mathcal{A}^{\mathcal{O}_\mathcal{S}}(\,)$
3. Return $g == b$.

At the beginning of the experiment, the challenger picks a random bit b. The adversary $\mathcal{A}$ subsequently interacts with the challenger by means of the following oracles $\mathcal{O}_\mathcal{S} = \{$`CreateReader`, `CreateTag`, `RegisterTag`, `Launch`, `DrawTag`, `Free`, `SendTag`, `SendReader`, `Result`, `Corrupt`, `CreateInsider`$\}$ and outputs a guess bit g.

According to the above experiment description, the challenger presents to the adversary the system where either the 'left' tags T_i (if $b = 0$) or the 'right' tags T_j (if $b = 1$) are selected when returning a virtual tag reference in `DrawTag`. The function $f_0 \in \mathcal{F}$ (where $\mathcal{F} = \{f : \{1, 2, \ldots, n\} \to \mathcal{T}\}$, see Sect. 5.2.4) maps the `DrawTag` invocations (referenced by an index k) to the tag T_i, which was passed as first argument to `DrawTag`. Similarly, f_1 maps invocation serial numbers to the second argument to `DrawTag`. f_0 and f_1 therefore describe the 'left' and the 'right' world, respectively.

$\mathcal{A}$ queries the oracles a number of times and, subsequently, outputs a guess bit g. We say that $\mathcal{A}$ wins the privacy game if and only if $g = b$, i.e. if it correctly identifies which of the worlds was active. The advantage of the adversary is defined as

$$\mathbf{Adv}_{\mathcal{S},\mathcal{A}}(k) = \left| Pr\left[\mathbf{Exp}^{b=0}_{\mathcal{S},\mathcal{A}}(k) = 1\right] + Pr\left[\mathbf{Exp}^{b=1}_{\mathcal{S},\mathcal{A}}(k) = 1\right] - 1\right| . \quad (5.1)$$

5.3.1 Privacy

The adversary restrictions, as defined in Sect. 5.2.1, also apply to our privacy definition. Depending on the acceptable usage of the `Corrupt` oracle, an adversary in our model is either strong, destructive (`Corrupt` destroys a tag), forward (after the first `Corrupt` only further corruptions are allowed), or weak (no `Corrupt` oracle) adversaries. Depending on the allowed usage of the `Result` oracle, there exist narrow (no `Result` oracle) and wide adversaries.

If an adversary is allowed to call `CreateInsider` the privacy notion is called '...-insider', so we can speak of forward-insider and weak-insider adversaries. For wide-strong and wide-destructive the `CreateInsider` can be simulated using the normal `CreateTag` and `Corrupt` oracles, i.e. strong-insider and destructive-insider are equivalent to wide-strong and wide-destructive respectively.

$$\begin{tikzcd}
 & & \text{forward-insider} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{weak-insider} \arrow[d, Rightarrow] \\
\text{wide-strong} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{wide-destructive} \arrow[r, Rightarrow] \arrow[ur, Rightarrow] \arrow[d, Rightarrow] & \text{wide-forward} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & \text{wide-weak} \arrow[d, Rightarrow] \\
\text{narrow-strong} \arrow[r, Rightarrow] & \text{narrow-destructive} \arrow[r, Rightarrow] & \text{narrow-forward} \arrow[r, Rightarrow] & \text{narrow-weak}
\end{tikzcd}$$

Definition 10 (Privacy)**.** *An RFID system $\mathcal{S}$, is said to unconditionally provide privacy notion X, if and only if for all adversaries $\mathcal{A}$ of type X, it holds that $\mathbf{Adv}^{X}_{\mathcal{S},\mathcal{A}}(k) = 0$. Similarly, we speak of computational privacy if for all polynomial time adversaries, $\mathbf{Adv}^{X}_{\mathcal{S},\mathcal{A}}(k) \leq \epsilon(k)$.*

We also define X^+ privacy notion variants, where X refers to the basic privacy notion and $+$ to the notion that arises when the corruption abilities of the adversary are further restricted with respect to the `DrawTag` oracle (see [25]). Formally, an RFID system is said to be X^+ private if it is X private and if, for all adversaries, $f_0 \approx_{\hat{T}} f_1$. Here, $f_0 \approx_{\hat{T}} f_1$ means that for all i such that $f_0(i) \in \hat{T}$ or $f_1(i) \in \hat{T}$, it holds that $f_0(i) = f_1(i)$, where $\hat{T}$ denotes the set of corrupted tags. This implies that, whenever a tag is corrupted at some point during the privacy game, it always has to be drawn simultaneously in both the left and the right world using a `DrawTag(`T_i, T_i`)` query with identical arguments.

Similarly, we define the X^* privacy notion variants that arise when the corruption abilities of the adversary are further restricted with respect to the `Corrupt` oracle. The restricted `Corrupt` oracle will only return the non-volatile state of the tag.

5.3.2 Security

Correctness ensures that a legitimate tag is always accepted by a reader.

Definition 11. *Correctness. A scheme is correct if the identification of a legitimate tag only fails with negligible probability.*

Soundness is the property that a fake tag is not accepted by the server. We only consider adversaries that cannot interact with the tag they try to impersonate during the identification protocol (i.e. we do not consider relay or concurrent attacks). To avoid relay attacks, distance bounding protocols can be deployed.

Rasmussen *et al.* [137] proposed an implementation of such a protocol with analog components that is suitable for RFID tags. The following definition differs from most models as we do not require matching conversations, but impersonation resistance as in [28] is sufficient.

Definition 12. *Soundness. A scheme is resistant against impersonation attacks if any polynomially bounded strong adversary cannot be identified by a verifier as the tag it impersonates, except with negligible probability. Adversaries may interact with the tag they want to impersonate prior to, and with all other tags prior to and during the protocol run. Nevertheless, the resulting protocol transcript must not lead to the identification of a corrupted tag. The adversary has no access to the `CorruptReader` oracle.*

Definition 13. *Extended Soundness. Identical to Def. 12, but the adversary is given access to the `CorruptReader` oracle.*

5.3.3 Modelling details

There are certain notable differences between our model and other models discussed in the previous section:

- The introduction of `CreateTag(·)`: since the set of tags is not predefined we allow the adversary to dynamically create new tags.
- `DrawTag(·,·)` and `Free(·)` are used to introduce the concept of virtual tags. This concept is needed since otherwise `SendTag(·,·)` would have to accept two tag/message pairs (and select one of them based on the value of b). In this case it would be trivial to determine the bit b for multi-pass protocols, simply by using different tags for each pass of the protocol if $b = 0$ and the same tag if $b = 1$. The protocol would only succeed if $b = 1$, thus allowing detection of b. Hence, it is crucial that the same tag is always used within a certain protocol run, which can be ensured by using virtual tag identifiers.
- A separate communication oracle for tags and reader is used, since the reader is not considered as an entity whose privacy can be compromised.
- `Corrupt(·)`: corruption is done with respect to a tag, not a virtual tag. If `Corrupt(·)` would accept a vtag, then determining the bit b becomes trivial by performing the following attack:

  - $vtag_a \leftarrow$ `DrawTag(`T_1`,`T_2`)`
  - $C_a \leftarrow$ `Corrupt(`$vtag_a$`)`
  - `Free(`$vtag_a$`)`
  - $vtag_b \leftarrow$ `DrawTag(`T_1`,`T_3`)`
  - $C_b \leftarrow$ `Corrupt(`$vtag_b$`)`

  If $C_a = C_b$ then $b = 0$, otherwise $b = 1$.

  We believe that it is realistic to assume that one has the tag identifier T_i when corrupting a tag, since corruption implies having physical access to a tag.

  Note that stateful protocols (which update their state after a protocol run) do not satisfy our privacy definition. By issuing a `Corrupt(`T_i`)` query before and after a protocol run, one can always identify whether or not the tag has been active. For such protocols, one could use the significantly weaker X^+ privacy notions.

- In the current setup `Corrupt(`T_i`)` reveals the full internal state of the tag, i.e. both its volatile and non-volatile parts. This follows from [6], where it is shown that, if corruptions reveal the volatile state, then the resulting privacy notions are stronger. Single-pass protocols (e.g. challenge-response) do not suffer from any issues, since the volatile memory is typically erased after sending the reply, and hence all computations are confined to the invocation of the `SendTag` oracle. Multi-pass protocols on the contrary, typically require storage of data in between `SendTag` invocations. Because corruption yields the entire internal state, one could make additional assumptions on the corruption abilities of the adversary by restricting corruption to the non-volatile state, using the use the X^* privacy notions. An even stronger restriction would be to allow only corruption of tags that are not drawn in either the left or right world; or use the X^+ privacy notions.

5.4 Previously Proposed Protocols

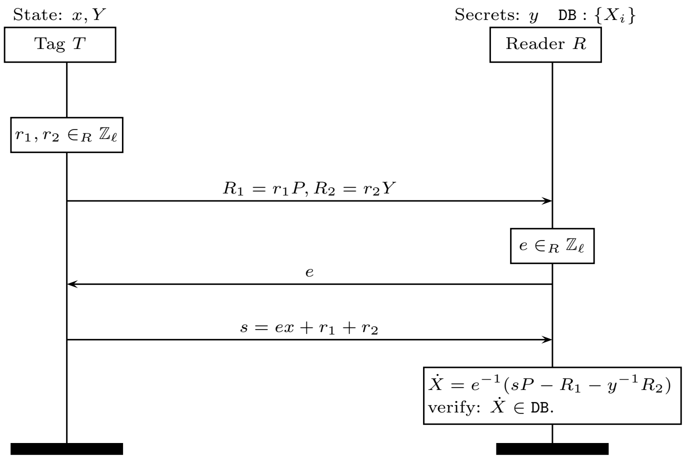
In this section, we give an overview of previously proposed protocols that are based on public key cryptography. Each of these protocols is correct, sound and achieves narrow-strong privacy.

5.4.1 Zero Knowledge Based Protocols

The zero knowledge based protocols are proofs of knowledge for a specific verifier (reader) with public key $Y = yP$. The prover (tag) proves knowledge of the private key $x \in \mathbb{Z}_\ell$, which is the discrete logarithm of the corresponding public key $X = xP$, for P a publicly agreed-on generator of $\mathbb{G}_\ell$. The public key X of the tag will serve as its identity and has been registered with the reader.

Zero knowledge based protocols are somewhat related to designated verifier proofs [91], where one proves that either some statement is true or one is the designated verifier. This setting is clearly different from private identification where one wants to protect the privacy of tags identifying to one reader (verifier). Laguillaumie and Vergnaud [102] proposed strong designated verifier signatures, based on the privacy of the signer's identity. In this setting any non-designated verifier cannot tell apart signatures from different signers to the same designated verifier. However, the designated verifier needs the identity of the signer to verify the signature.

Randomised Schnorr: was proposed by Bringer *et al.* [27] (see Fig. 5.2). It achieves extended soundness and narrow-strong* privacy. This protocol requires only two scalar-EC point multiplications at the tag side.



State: x, Y
Tag T
Secrets: y DB : $\{X_i\}$
Reader R
$r_1, r_2 \in_R \mathbb{Z}_\ell$
$R_1 = r_1P, R_2 = r_2Y$
$e \in_R \mathbb{Z}_\ell$
e
$s = ex + r_1 + r_2$
$\dot{X} = e^{-1}(sP - R_1 - y^{-1}R_2)$
verify: $\dot{X} \in$ DB.

Figure 5.2: Randomised Schnorr [27].

Randomised Hashed GPS: was later proposed by Bringer *et al.* [28] (see Fig. 5.3). The protocol has extended soundness and narrow-strong* privacy. The authors also claim wide-PI-forward* privacy, i.e. wide-forward* privacy even when the list of registered tags' identities is known. This approach requires two scalar-EC point multiplications and the evaluation of a hash function, for which additional hardware will be needed.

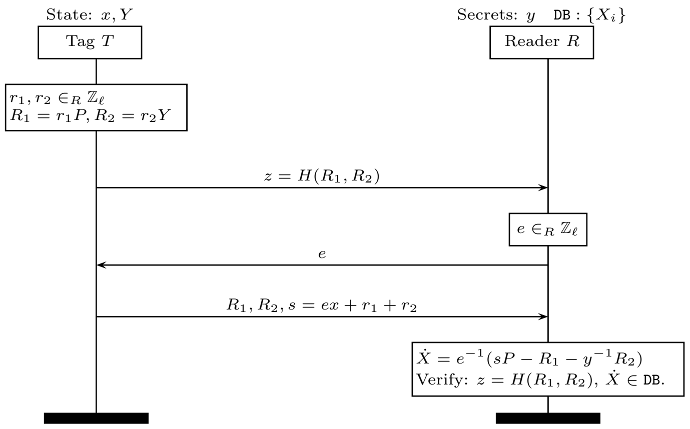

State: x, Y
Tag T
Secrets: y DB : $\{X_i\}$
Reader R
$r_1, r_2 \in_R \mathbb{Z}_\ell$
$R_1 = r_1P, R_2 = r_2Y$
$z = H(R_1, R_2)$
$e \in_R \mathbb{Z}_\ell$
e
$R_1, R_2, s = ex + r_1 + r_2$
$\dot{X} = e^{-1}(sP - R_1 - y^{-1}R_2)$
Verify: $z = H(R_1, R_2)$, $\dot{X} \in$ DB.

Figure 5.3: Randomised Hashed GPS [28].

Privacy-wise, both protocols suffer from the adversary having complete freedom over the exam e it sends to the tag and the fact that the final message from the tag s contains a term that is linearly dependent on this exam and the secret of the tag x. For this reason these protocols cannot be wide-strong private[5].

Furthermore, there exist a linear relation between the commitments (R_1, R_2) and the answer s. This, together with the above, makes that Randomised Schnorr cannot be wide-weak private[6]. Both protocols are also vulnerable to insider-attacks[7].

[5]An attacker in the middle sends $e - 1$ to the virtual tag and responds to the reader with $s + x$. For a correct guess of the tag's identity with known internal state x, the `result` oracle returns 1.

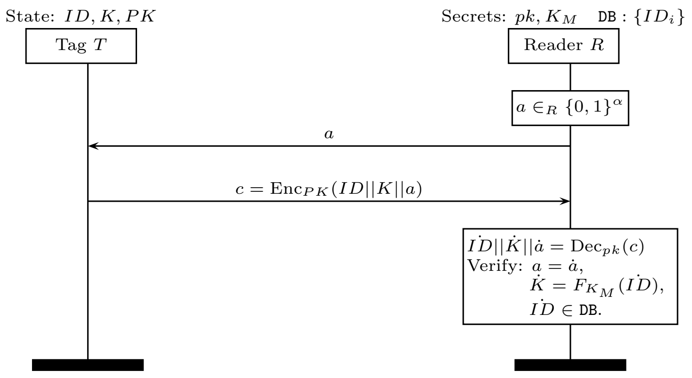
[6]For an observed protocol run π_0, an adversary can test, using the `result` oracle, that the current virtual tag is the tag of π_0. The adversary mounts a man-in-the-middle attack, sends to the reader $(R_1 + R_{1,0}, R_2 + R_{2,0})$, challenges the tag with $e - e_0$ and returns to the reader $s + s_0$.

[7]Similar to the above. The attacker sends the exam e_0 to the virtual tag in protocol run π_1. When subtracting the answers $s_0 - s_1$, the tag specific part should cancel out. The attacker starts a protocol run π_2 between its insider tag (with private key x') and the reader. The attacker sets $R_1 = R_{1,0} - R_{1,1}$, $R_2 = R_{2,0} - R_{2,1}$ and replies with $s' = s_0 - s_1 + e_2x'$.

5.4.2 Public Key Encryption Based Protocols

For these protocols, the reader has a public/private key pair (PK, pk). The identities ID of tags that registered, are stored in the reader's database. The tag and reader share a symmetric key K.

Vaudenay's Public Key Protocol: [162] (see Fig. 5.4) requires the tag to compute the public key encryption of one message. This cryptosystem needs to be secure against adaptive chosen ciphertext attacks (IND-CCA2) to have a secure identification scheme that achieves narrow-strong and wide-forward privacy. As shown by Hermans *et al.* [84], this protocol achieves wide-strong privacy. One of the most efficient IND-CCA2 cryptosystems in the standard model is the Cramer-Shoup cryptosystem [43]. This cryptosystem requires five scalar-EC point multiplications, two EC point additions and one evaluation of a hash function for one encryption.



State: ID, K, PK
Tag T
Secrets: pk, K_M $\mathtt{DB} : \{ID_i\}$
Reader R
$a \in_R \{0,1\}^\alpha$
a
$c = \mathrm{Enc}_{PK}(ID||K||a)$
$\dot{ID}||\dot{K}||\dot{a} = \mathrm{Dec}_{pk}(c)$
Verify: $a = \dot{a}$,
$\dot{K} = F_{K_M}(\dot{ID})$,
$\dot{ID} \in \mathtt{DB}$.

Figure 5.4: Vaudenay's Public Key Protocol [162].

Hash ElGamal-Based Protocol was proposed by Canard *et al.* [33] (see Fig. 5.5). This protocol is secure, narrow-strong private and future untraceable. It is unclear how future untraceability (as defined by Canard *et al.* [33]) and wide-strong privacy relate to each other; however, these seem to be closely related. This protocol is more efficient than Vaudenay's public key protocol. It uses a cryptosystem that is secure against chosen plaintext attacks (IND-CPA), Hash ElGamal, and a MAC algorithm. This scheme is more efficient than Vaudenay's public key scheme since the underlying encryption does not need to be IND-CCA2. Note that the combination of a MAC and IND-CPA encryption used in this specific protocol in fact provides IND-CCA2 encryption for the type of plaintext messages used [101]. The tag is required to compute two scalar-EC point multiplications, one evaluation of a hash function and one evaluation of a MAC algorithm.

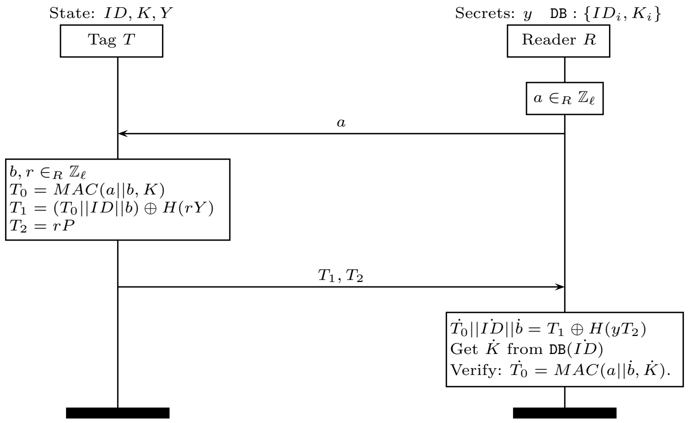

State: ID, K, Y
Tag T
Secrets: y DB : $\{ID_i, K_i\}$
Reader R
$a \in_R \mathbb{Z}_\ell$
a
$b, r \in_R \mathbb{Z}_\ell$
$T_0 = MAC(a||b, K)$
$T_1 = (T_0||ID||b) \oplus H(rY)$
$T_2 = rP$
T_1, T_2
$\dot{T_0}||\dot{ID}||\dot{b} = T_1 \oplus H(yT_2)$
Get $\dot{K}$ from DB($\dot{ID}$)
Verify: $\dot{T_0} = MAC(a||\dot{b}, \dot{K})$.

Figure 5.5: Hash ElGamal-Based Protocol [33].

Neither protocol achieves extended soundness. The tag and the reader need to store some shared (secret) data. These shared data consist of an identifier ID and a shared secret key K. Both protocols achieve wide-strong privacy and soundness can also be proven under the more strict definition of matching conversations. Wide-strong privacy also rules out insider attacks on privacy.

5.5 A New Protocol

Our proposed protocol is a modified version of the Schnorr identification protocol [143]. The original protocol is proven secure by Bellare [19] under the One More Discrete Logarithm assumption (cf. Sect. 1.5.3). This protocol consists three passes: commit, exam and response. A consequence of having a three pass protocol is that only the X* privacy notions can be reached.

Our starting point is a variant of the Schnorr identification protocol, where the exam of the reader is applied to the tag's randomness instead of its secret. This variant is equivalent to the original protocol, except for the case $e = 0$. In the original Schnorr identification protocol this results in the adversary learning the tag's randomness while in the variant the adversary will learn the tag's secret. This situation can easily be avoided by having the tag check that $e \neq 0$.

Privacy is ensured by introducing a blinding factor d that can only be computed by the tag and the reader. The blinding factor is applied to the secret x of the tag in its response. This blinding factor only depends on input of the tag and the public key of the reader, which is known to the tag. Hence an adversary cannot influence the value of this blinding factor. In contrast to previously proposed zero-knowledge based protocols (see Sect. 5.4.1), the factor applied to the secret cannot be influenced by an adversary.

An overview of the proposed protocol is given in Fig. 5.6. The tag generates two random numbers r_1 and r_2, where the former is needed for extended soundness and the latter is used to ensure privacy. The tag commits to its randomness by sending R_1, R_2 to the reader. The reader verifies that $R_2 \neq O$, the point at infinity. The tag's response is $s = dx + er_1$, with d the blinding factor as computed by the tag. The reader verifies by checking that a tag with public key $\dot{X} = \dot{d}^{-1}(sP - eR_1)$, with $\dot{d}$ the blinding factor as computed by the reader, has been registered. The reader keeps a list of all incomplete sessions. If a session timeout occurs or the tag fails to identify for a given challenge, the session is also considered to be completed.

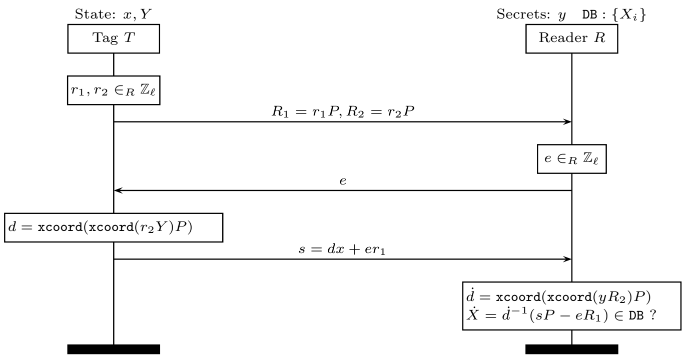

State: x, Y
Tag T
Secrets: y DB : $\{X_i\}$
Reader R
$r_1, r_2 \in_R \mathbb{Z}_\ell$
$R_1 = r_1P, R_2 = r_2P$
$e \in_R \mathbb{Z}_\ell$
e
$d = \texttt{xcoord}(\texttt{xcoord}(r_2Y)P)$
$s = dx + er_1$
$\dot{d} = \texttt{xcoord}(\texttt{xcoord}(yR_2)P)$
$\dot{X} = \dot{d}^{-1}(sP - eR_1) \in \texttt{DB}$?

Figure 5.6: Private RFID Identification Protocol.

The blinding factor contains $r_2Y = yR_2$. Given the Compuational Diffie-Hellman assumption (cf. Sect. 1.5.3), this value can only be computed when given either r_2 or y. To prevent an adversary of exploiting the self-reduciblity of the Discrete Logarithm problem (cf. Sect. 1.5.3), this value is encapsulated by applying a one-way function. An obvious one-way function is a cryptographic hash function. However, to implement a cryptographic hash function on an RFID tag, additional logic is required. Current hash functions [147] require at least 50% of the circuit area of the most compact ECC implementation. For this reason we propose the following one-way function, that is built using only EC operations $H(r_2Y) = \mathtt{xcoord}(r_2Y)P$. The value d is set to the x-coordinate of the EC point.

5.5.1 Analysis

The first two theorems deal with the security properties, correctness and extended soundness, of the proposed protocol. The last theorem covers the protocol's privacy properties.

Theorem 5. *The proposed protocol is correct according to Def. 11.*

Proof. Since $d = \mathtt{xcoord}(\mathtt{xcoord}(r_2Y)P) = \mathtt{xcoord}(\mathtt{xcoord}(r_2yP)P)$
$= \mathtt{xcoord}(\mathtt{xcoord}(yR_2)P) = \dot{d}$,
it follows that $\dot{X} = \dot{d}^{-1}(sP - eR_1) = d^{-1}((dx + er_1)P - er_1P) = X$. □

Theorem 6. *The proposed protocol has extended soundness according to Def. 12 under the OMDL assumption.*

Proof. Assume an adversary $\mathcal{A}$ that can break the extended soundness with non-negligible probability, i.e. that can perform a fresh, valid authentication with the verifier. Without loss of generality we will assume the target tag is known at the start of the game[8]. We construct an adversary $\mathcal{B}$ that wins the OMDL game as follows:

- Set $X = \mathcal{O}_2()$. X will be used as the public key of the target tag.
- $\mathcal{B}$ executes $\mathcal{A}$. During the first phase of $\mathcal{A}$, $\mathcal{B}$ simulates the SendTag oracles for the target tag as follows (all other oracles are simulated as per protocol specification):
    - On the first $\mathtt{SendTag}(vtag)$ query of the i'th protocol run: return $R_{2,i} = r_{2,i}P$ with $r_{2,i} \in_R \mathbb{Z}_\ell$ and $R_{1,i} = \mathcal{O}_2()$.
    - On the second $\mathtt{SendTag}(vtag, e_i)$ query of the i'th protocol run: set $d_i = \mathtt{xcoord}(\mathtt{xcoord}(r_{2,i}Y)P)$ and return $s_i = \mathcal{O}_1(d_iX + e_iR_{1,i})$.

[8]Otherwise, the proof can be adapted by choosing the public key of the first tag as $X_1 = \mathcal{O}_2()$ and for all following tags $X_i = r_iX_1$ with $r_i \in_R \mathbb{Z}_\ell$.

- During the second phase of $\mathcal{A}$, $\mathcal{B}$ proceeds as follows:
  - On the first call to $\texttt{Result}(\pi)$, compute $d = \texttt{xcoord}(\texttt{xcoord}(yR_2)P)$ and store (s, d). Next, rewind $\mathcal{A}$ until right before the call to $\texttt{SendReader}(\pi, R_1, R_2)$. On the next call to $\texttt{SendReader}(\pi, R_1, R_2)$, return a new random e'.
  - On the next call to $\texttt{Result}(\pi)$: compute $r_1 = {}^{s-s'}/_{e-e'}$ and $x = d^{-1}(s - r_1)$ return $(x, e_1^{-1}(s_1 - d_1 x), \ldots, e_k^{-1}(s_k - d_k x))$.

The simulation by $\mathcal{B}$ is perfect during both phases. At the end of the game $\mathcal{B}$ will successfully win the OMDL with non-negligible probability, unless $s = s'$, which happens with negligible probability since both e and e' are randomly chosen after $R_2 \neq O$ is fixed. □

Privacy of the protocol is proven under the Oracle Diffie-Hellman (ODH) and x-Logarithm (XL) assumptions (cf. Sect. 1.5.3).

Theorem 7. *The proposed protocol is wide-strong* private according to Def. 10 under the ODH and the XL assumption.*

Proof. Assume an adversary $\mathcal{A}$ that wins the privacy game with non-negligible advantage. Using a standard hybrid argument [173, 74], we construct an adversary that breaks the ODH-assumption. We set $Y = B$. $\mathcal{B}_i$ plays the privacy game with $\mathcal{A}$. $\mathcal{B}_i$ selects a random bit $\tilde{b}$ that indicates which world is simulated to $\mathcal{A}$. All oracles are simulated in the regular way, with the exception of the `SendTag` and `Result` oracle for the target tag:

- $\texttt{SendTag}(vtag)$:
  - $j \neq i$: Generate $r_1, r_2 \in_R \mathbb{Z}_\ell$. Take $R_1 = r_1 P, R_2 = r_2 P$. Return R_1 and R_2.
  - $j = i$: Generate $r_1 \in_R \mathbb{Z}_\ell$.Take $R_1 = r_1 P, R_2 = A$. Return R_1 and R_2.
- $\texttt{SendTag}(vtag, e)$, j'th query: retrieve the tuple $(vtag, T_0, T_1)$ from the table $\mathcal{D}$. Take the key x for tag $T_{\tilde{b}}$.
  - $j < i$: Generate $r \in_R \mathbb{Z}_\ell$. Take $d = \texttt{xcoord}(H(rP))$. Return $s = dx + er_1$.
  - $j = i$: Take $d = \texttt{xcoord}(\mathcal{O}_2())$. Return $s = dx + er_1$.
  - $j > i$: Take $d = \texttt{xcoord}(H(r_2 Y))$. Return $s = dx + er_1$.

- `Result`(π): If the received R_2 in session π matches A from the ODH problem take $\dot{d} = \texttt{xcoord}(\mathcal{O}_2())$. If not, check if R_2 matches any of the R_2's generated during the first $i-1$ `SendTag` queries. If so, use the r generated in that query and compute $\dot{d} = \texttt{xcoord}(H(rP))$. Otherwise, take $\dot{d} = \texttt{xcoord}(\mathcal{O}_1(R_2))$. Finally, compute $\dot{X} = \dot{d}^{-1}(sP - eR_1)$. Check $\dot{X}$ with the database, return true if $\dot{X}$ is found, false otherwise.

At the end of the game $\mathcal{A}$ outputs its guess g for the privacy game. $\mathcal{B}_i$ outputs $(\tilde{b} == g)$.

The above simulation to $\mathcal{A}$ is perfect, since validation is done in the same way as the protocol specification. If $R_2 = A$, the oracle $\mathcal{O}_1(\cdot)$ cannot be used. However, in this case we know the corresponding value of d by directly calling the $\mathcal{O}_2()$ oracle, which gives the same result.

We use $\mathcal{A}^i$ (with $i \in [1 \ldots k]$) to denote the case that $\mathcal{A}$ runs with the first i `SendTag` queries random instances, and the other queries real instances. This is the case when $\mathcal{B}_{i+1}$ runs with a real ODH instance, or $\mathcal{B}_i$ with a random ODH instance.

By the hybrid argument we get:

$$|| \Pr\left[\mathcal{A}^0\text{wins}\right] - \Pr\left[\mathcal{A}^k\text{wins}\right] || \leq \sum \mathbf{Adv}_{B_i}\,.$$

Note that $\mathcal{A}^i$ wins if $\tilde{b} == g$. In the case of $\mathcal{A}^0$, $\Pr\left[\mathcal{A}^0\text{wins}\right] = \Pr\left[\mathcal{A}\text{wins}\right]$. In the case of $\mathcal{A}^k$ however, the output from `SendTag` is always random and independent of x: note that $d = \texttt{xcoord}(\texttt{xcoord}(rP)P)$ with $r \in_R \mathbb{Z}_\ell$. Under the XL assumption it follows that d is uniformly random and independent of R_1, R_2 and e. Since $s = dx + er_1$, s is also uniformly random and independent of R_1, R_2, e and x. So $\mathcal{A}^k$ has probability $1/2$ of winning the privacy game, since it obtains no information at all from a tag.

$$\begin{aligned}
|| \Pr\left[\mathcal{A}^0\,\text{wins}\right] - \Pr\left[\mathcal{A}^k\,\text{wins}\right] || &= || \Pr\left[\mathcal{A}\,\text{wins}\right] - \frac{1}{2} || \\
&= \frac{1}{2}\mathbf{Adv}_{\mathcal{A}}^{privacy} \\
&\leq \sum \mathbf{Adv}_{B_i}\,.
\end{aligned}$$

It follows that at least one of the $\mathcal{B}_i$ has non-negligible probability to win the ODH game. □

5.5.2 Efficiency Optimisation

Only one random value r is generated by the tag ($r_1 = r_2$). Hence, the tag has to compute one less scalar-EC point multiplication and has to transmit one less element. The blinding factor is changed to $d = \texttt{xcoord}(r_2 Y)$. This reduces the computational effort required from the tag with another scalar-EC point multiplication. The function to compute the blinding factor is no longer one-way for $r_2 Y$, however, the response s is. An overview of the protocol is given in Fig. 5.7

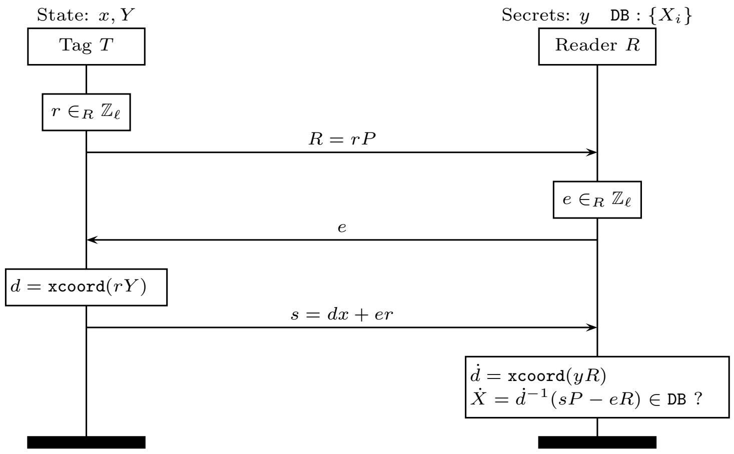

State: x, Y
Tag T
Secrets: y DB : $\{X_i\}$
Reader R
$r \in_R \mathbb{Z}_\ell$
$R = rP$
$e \in_R \mathbb{Z}_\ell$
e
$d = \texttt{xcoord}(rY)$
$s = dx + er$
$\dot{d} = \texttt{xcoord}(yR)$
$\dot{X} = \dot{d}^{-1}(sP - eR) \in \texttt{DB}$?

Figure 5.7: Optimised Private RFID identification protocol.

Theorem 8. *The optimised protocol has extended soundness according to Def. 13 under the OMDL assumption.*

Proof. The proof is the same as the proof for the basic version of the protocol, except that $d = \texttt{xcoord}(rY)$ and $R = R_1 = R_2$. □

For privacy an extended ODH variant is required with some additional oracles. The original ODH variant from Def. 1 gives direct access to an oracle for computing the blinding factor d. This is no longer possible since $d = \texttt{xcoord}(C)$ does not involve a one-way function and allows recovery of C.

Theorem 9. *The optimised protocol is wide-strong* private according to Def. 10 under an extended ODH assumption.*

The privacy of the optimised protocol can be shown under a modified ODH assumption featuring the following oracles:

- $\mathcal{O}_1(Z_1, Z_2) = \texttt{xcoord}(bZ_1)^{-1}Z_2$ with $Z_1 \neq \pm A$
- $\mathcal{O}_2(Z) = \texttt{xcoord}(C)^{-1}Z$
- $\mathcal{O}_3(z) = \texttt{xcoord}(C)z + a$ (can only be invoked once)

The adversary is given access to these three oracles and $A = aP, B = bP$. Note that $\mathcal{O}_3$ can only be called once during the ODH game to prevent the value a from being leaked, while the other oracles can be called multiple times.

A similar privacy proof as in Sect. 5.5.1 can be used, with different oracle calls in the `SendTag` and `Result` simulation. In this case $e \cdot \mathcal{O}_3(e^{-1}x)$ is used in `SendTag` for generating a reply (if $i = j$) and $\mathcal{O}_1$ and $\mathcal{O}_2$ are used in the `Result` oracle to replace the computation of $d^{-1}Q$ with $Q = sP - eR_1$.

5.6 Implementation Considerations

Our protocols require the evaluation of scalar-EC point multiplications and the generation of a random number. For 80-bit security, we need an elliptic curve over a field that is approximately 160 bits in size. The protocol can be implemented on the architecture proposed by Lee *et al.* [107]. Their ECC coprocessor can be built with less than 15 kGEs (Gate Equivalent), consumes about 13.8 μW and takes around 85 ms for one scalar-EC point multiplication. More recently, Wenger and Hutter [169] proposed an ECC coprocessor that only requires 9 kGEs, consumes about 32.3 μW and takes around 286 ms for one scalar-EC point multiplication. Aside from the ECC coprocessor, circuit area is required for the ROM (Read Only Memory), RAM (Random Access Memory) and RNG (Random Number Generator).

5.6.1 (Non-)Sense of Coupons

Several papers [28, 33] proposed to optimise their private RFID authentication protocols by means of precomputation. The protocol is split into an off-line and on-line phase, for which less computational effort is required in the on-line phase. Hence, the protocol can be executed faster. The precomputed values are stored in the form of coupons. There are two ways of implementing these coupons: either these coupons are precomputed externally and pushed on the tag or the tag generates these coupons itself.

First we will discuss the former. This way has the additional benefit that for most protocols, less logic needs to be implemented on the tag. Protocols that only need EC point additions in the on-line phase, compared to a full fledged EC coprocessor, allow to save a lot of circuit area[9]. The most striking example is the case of Randomised Hashed GPS [28] for which, when using coupons, the only required logic is a hardware implementation of scalar arithmetic. This in contrast to the original protocol, that requires additional logic to compute scalar-EC point multiplications and evaluate a hash function. The downside of the tag not being able itself to do these necessary computations is that an adversary can quite easily mount a Denial of Service (DoS) attack, by tricking the tag into authenticating over and over again until it has no coupons left. This can be prevented by introducing mutual authentication, more specifically have the reader first authenticate to the tag. Ironically (as shown in Sect. 5.7.2), the only efficient way to achieve authentication of the reader to a yet-unknown tag, is by using zero knowledge proofs, which in turn require a full fledged EC coprocessor on the tag to verify these. How to securely push these coupons onto the tag is an additional issue.

Having the tag precompute coupons can speed up the identification process, or alternatively make it possible to use a smaller but slower EC coprocessor. The tag computes these coupons whenever energy is available; tags can draw energy as long as these are in the electromagnetic field around any reader. Since the tag can do all the necessary computations itself, one only needs limited storage for coupons, namely a buffer of size b.

The disadvantage of coupons is that these need to be stored on the tag. When making abstraction of the control logic needed to access this storage, one still needs about one floating gate per bit[10]. As introduced by Girault *et al.* [73], the size of the coupons can be minimised when not storing the used randomness and instead implementing a pseudo-random function with a seed to generate random numbers on the tag. Taking this optimisation into account, the protocols discussed in this chapter still require coupons that consist of two elements. This means that, for the same circuit area, one can either implement the EC coprosessor (as proposed by Wenger and Hutter) or storage that allows for 20-30 coupons.

In general it can be argued that strong privacy is not achievable when using coupons or when using a pseudo-random function instead of a true random number generator. By making a query to the `Corrupt` oracle, the adversary learns the complete internal state of the tag, which also comprises coupons and/or the seed of the pseudo-random function. When the coupons are generated by the tag itself, b-strong privacy is possible, meaning that the tag becomes

[9]From Batina *et al.* [14], the hardware required for computing EC point additions is expected to take a few thousand GEs.

[10]This means that one can only store 6-7 elements for a circuit area equivalent to 1 kGE.

unlinkable again after the first b conversations from the moment the tag was freed after corruption.

Part of the coupons is reader specific. This puts an additional burden on tag storage requirements in the multi-reader setting, making the use of coupons completely impractical. For all these reasons, we do not consider coupons.

5.6.2 Comparison

Now we will compare our protocol and its optimised variant to previously proposed protocols, described in Sect. 5.4. A general overview of the protocols is given in Table 5.1.

Both the Randomised Schnorr and our proposed protocol benefit from a compact hardware design, only an ECC coprocessor is needed. The other protocols require additional hardware to evaluate a cryptographic hash function, which makes the design substantially larger[11].

The scalar-EC point multiplication is the most complex operation, followed by an EC point addition, and lastly the evaluation of a hash/MAC algorithm. For a fair comparison between the performance of protocols that require the evaluation of a hash/MAC and protocols that do not, we assume the same total available circuit size. This means that our protocol can be implemented using a larger but faster ECC processor.

When also considering the more general setting, where a single tag can identify the end-user privately to multiple readers, the tags not only need to store an extra public key for every reader but also corresponding shared data, if any. In this setting there is a clear advantage for protocols that provide extended soundness, since the tag can use the same private/public key pair to identify to each reader.

[11]Recall that current cryptographic hash functions [147] require at least 50% of the circuit area of the most compact ECC implementation.

Table 5.1: Overview Private RFID Identification Protocols.

Protocol	Strongest Privacy	Insider Private	Extended Soundness	R_i-specific state	Operations
Randomised Schnorr [27]	narrow-strong*	no	yes	Y_i	2 EC mult
Randomised Hashed GPS [28]	narrow-strong* wide-forward*	no	yes	Y_i	2 EC mult 1 hash
Vaudenay [162] + Cramer-Shoup [43]	wide-strong	yes	no	K_i C_i, D_i, H_i	5 EC mult 2 EC add 1 hash
Hash ElGamal [33]	wide-strong	yes	no	K_i Y_i	2 EC mult 1 hash 1 MAC
Proposed Protocol (Sect. 5.5)	wide-strong*	yes	yes	Y_i	4 EC mult
- Optimised version (Sect. 5.5.2)	wide-strong*	yes	yes	Y_i	2 EC mult

5.7 Mutual Authentication

One can wonder if mutual authentication is a useful feature for private RFID authentication protocols. This raises the following question: Will the additional effort required from the tag result safer or more private protocols? At first glance, yes. One of the issues of RFID tags is that these will respond to any query. When an RFID tag only responds to authorised queries, the attacker's power is reduced. Mutual authentication is also a prerequisite to establish a secure connection, which in term allows to exchange data privately.

Paise and Vaudenay [124] proposed the only existing private RFID mutual authentication model, which is an extension of Vaudenay's private RFID authentication model [162]. Their motivation for this model is to solve the issue mentioned above, that can be fixed by requiring that "a tag must be confident of the reader's identity before sending any information or its ID." However, mutual authentication is modelled in the complete opposite way, i.e. by enriching protocols with an extra message from the reader to the tag. Technically mutual authentication is achieved, but by having the tag first authenticate to the reader, before the reader authenticates to the tag, the problem remains. RFID tags will still respond with identifiable information to unauthorised queries. A second problem with this way of modelling is that the tag is expected to output a bit, indicating whether or not mutual authentication was successful. Oddly enough, the adversary is not given access to an oracle that allows to learn this outcome. Furthermore, the assumption that the tag outputs this bit is not realistic as most RFID tags, even smart cards, do not have a user interface. This means that it is impossible for the end-user of the RFID tag to spot the difference between a successful mutual authentication or the protocol being aborted by the tag. We can conclude that private RFID mutual authentication under this model, requires additional effort from the RFID tag and does not result in stronger security or privacy guarantees when the goal is merely authentication.

An exception is the class of private RFID authentication protocols that rely on the tag and server to keep a synchronised state, which is updated after every instance of the authentication protocol. The tag will update his state and send some information allowing the server to compute the same state. To avoid desynchronisation attacks, the server first confirms the new state to the tag (and as a consequence, implicitly authenticates to the tag), before the tag updates its state. For these protocols, mutual authentication results in stronger security guarantees. However, this class is explicitly excluded in the Paise-Vaudenay model [124] as they only consider enriched protocols where "the final message from the reader to the tag does not modify the tag state."

When the reader authenticates first to the tag, tag privacy might be enhanced. A good example is the class of private search protocols [176], sometimes also referred to as private interrogation protocols [26]. In these protocols, the reader wants to know if a certain RFID tag is in the neighbourhood in a privacy-preserving way. These protocols are designed such that only the target RFID responds. Having the server first authenticate to the tag prevents adversaries from tracing tags. However, these protocols only provide mutual authentication for one specific tag and do not consider the more general setting, where the reader needs to prove its identity to non-designated verifier.

We extend our model to include mutual authentication. We explicitly define mutual authentication as the reader first authenticating to the yet-unknown tag, the tag authenticating to the reader and the coupling between them. By doing authentication in this order it is easier to achieve private protocols: the tag is certain of the identity of the reader before revealing its own identity. Moreover it allows for narrow protocols, where the reader does not give any output on success or failure of the tag authentication, whereas this would be impossible if tag authentication preceeds reader authentication. As an additional benefit of preceeding tag authentication with reader authentiation, the end-user of the RFID tag to learn the outcome of the protocol, since the reader, in contrast to the tag, has a user interface. When mutual authentication is only intended to set up secure communication, the order of authentication between tag and reader is less important.

One can construct RFID tags that are only to be used for a limited number of authentication instances, by only storing this number of coupons (see Sect. 5.6.1). Without mutual authentication (where the server authenticates first), an adversary can mount a very simple DoS attack.

5.7.1 Model

Tag-authentication is defined by tag-correctness and tag-soundness. Similarly, reader-authentication is defined by reader-correctness and reader-soundness. Mutual authentication is achieved when both tag-authentication and reader-authentication took place in one and the same session. Towards this goal we define mutual authentication by redefining tag-authentication.

Definition 14. *Mutual Authentication. Tag authentication implies that reader authentication was achieved in the same session, otherwise the tag will abort.*

5.7.2 Protocol

Since the reader needs to authenticate to a non-designated party, authentication based on a mutual secret is ruled out. Instead the reader needs to provide a proof of knowledge of its secret key when challenged by the yet-unknown tag.

Since we only consider tag-privacy and not the reader's, mutual authentication can be achieved by prepending an existing private RFID authentication with an instance of the generic Schnorr identification [143] and binding these two authentications. Alternatively the reader can sign his challenge together with the input of the tag. Both general approaches result in additional computation effort required from the tag of at least two scalar-EC point multiplications.

To achieve mutual authentication, the protocol needs at least three rounds, where the tag needs to store information from the first round to be used later on in the third round. This means that, in order to achieve privacy notions where the adversary is allowed to query the `corrupt` oracle (i.e. forward and strong), restrictions need to be imposed on this oracle such that only the non-volatile state is returned. Instead of prepending existing two-round protocols where this restriction on the `corrupt` is not needed, we look at how to improve the more efficient three-round protocols, where this restriction already applied.

By having the reader sign its exam and the previous message in the Randomised Schnorr protocol, mutual authentication is achieved. Furthermore, the resulting protocol (as depicted in Fig. 5.8) is wide-forward private. This follows from Vaudenay's lemma [162, Lemma 8] stating that for a secure scheme, narrow-forward privacy implies wide-forward privacy. His definition of security, namely the tag should only be accepted for a matching conversation between tag and reader, is more strict than ours. However, by having an authenticated challenge, we also meet this more strict definition of security. Even though the attacks as described in Sect. 5.4.1 are precluded by requiring matching conversations, the insider-attacks described by van Deursen and Radomirović [160] are still possible. This means that wide-forward privacy is the strongest attainable privacy notion.

To achieve efficient mutual authentication and wide-strong* privacy, we transform our proposed protocol into a new protocol, IBIHOP. As such, we keep the strategy of hiding away the exam from unauthorised parties.

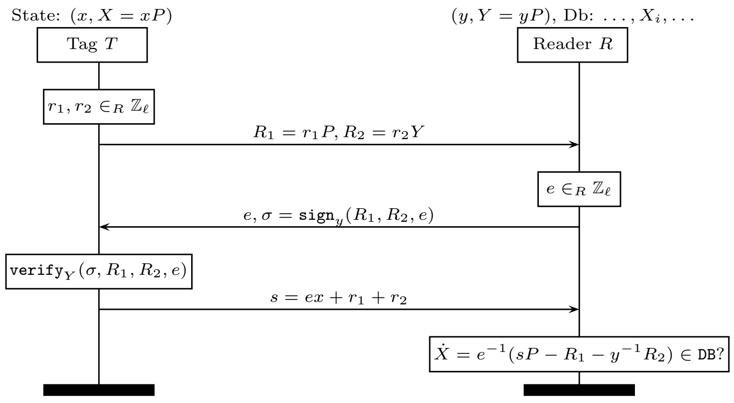

State: $(x, X = xP)$
$(y, Y = yP)$, Db: $\dots, X_i, \dots$
Tag T
Reader R
$r_1, r_2 \in_R \mathbb{Z}_\ell$
$R_1 = r_1 P, R_2 = r_2 Y$
$e \in_R \mathbb{Z}_\ell$
$e, \sigma = \mathtt{sign}_y(R_1, R_2, e)$
$\mathtt{verify}_Y(\sigma, R_1, R_2, e)$
$s = ex + r_1 + r_2$
$\dot{X} = e^{-1}(sP - R_1 - y^{-1}R_2) \in \mathtt{DB}?$

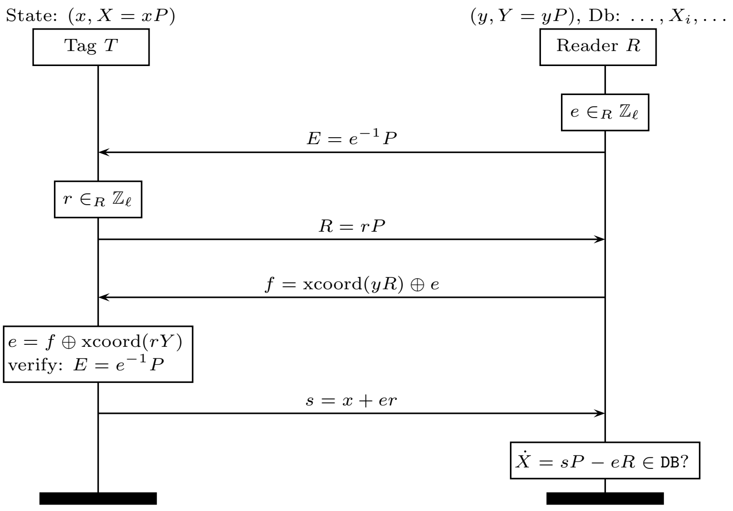
Figure 5.8: Mutual Authentication Randomised Schnorr.

IBIHOP

Our proposed mutual authentication wide-strong* private protocol, IBIHOP, is shown in Fig. 5.9. This proposal is very efficient as it only requires three scalar-EC point multiplications from the tag. Compared to our most efficient variant that required two scalar-EC point multiplications and one EC point addition, mutual authentication is achieved for only a slight increase in the required tag's effort.

The most crucial design decision for mutual authentication protocols is how to link two uni-directional authentication instances. Recall that one of our design criteria is to have a small hardware footprint, as such, a hash function is ruled out. In practice this also rules out signature schemes, as the message to be signed is hashed. We opted to interleave two interactive authentication protocols, resulting in a four pass protocol. Tag authentication is very similar to the optimised protocol. However, since only the tag can compute the exam e, which is known to the reader, from the broadcasted messages, there is no need for an additional blinding factor. For reader authentication, we have the reader first commit to the exam and then challenge it to prove knowledge of this value. This is a zero-knowledge proof under the CDH assumption, making it possible for the tag, having knowledge of r, to extract e. As such reader-authentication is achieved at the same time as providing the crucial coupling between the two authentication instances.

For the security proof, a computational variant of the oracle Diffie-Hellman assumption,denoted by OCDH, is required.



State: $(x, X = xP)$
Tag T
$(y, Y = yP)$, Db: $\ldots, X_i, \ldots$
Reader R
$e \in_R \mathbb{Z}_\ell$
$E = e^{-1}P$
$r \in_R \mathbb{Z}_\ell$
$R = rP$
$f = \text{xcoord}(yR) \oplus e$
$e = f \oplus \text{xcoord}(rY)$
verify: $E = e^{-1}P$
$s = x + er$
$\dot{X} = sP - eR \in \mathtt{DB}?$

Figure 5.9: IBIHOP.

Theorem 10. *IBIHOP achieves mutual authentication according to Def. 14 under the OCDH and the OMDL assumptions.*

Reader-authentication[12] of IBIHOP can be shown under a modified OCDH assumption.

An adversary has to ouput a pair $\{E, f\}$ for which $E = (f \oplus xcoord(abP))^{-1}P$ holds, given $A = aP, B = bP$ and the following oracles:

- $\mathcal{O}_1 = h$ (a handle h is returned to the value $e_h \in_R \mathbb{Z}_\ell$)
- $\mathcal{O}_2(Z, h) = \mathtt{xcoord}(bZ) \oplus e_h$ with $Z \neq \pm A$(only to be invoked once per h)
- $\mathcal{O}_3(Z, h) = e_h^{-1}Z$

From the reader-authentication one learns the value e, that is used as exam for the tag-authentication in the same session. At this point, only tag-authentication on its own remains to be shown. This can be be shown under the OMDL assumption.

[12]We only consider soundness, as correctness is trivial.

Theorem 11. *IBIHOP is wide-strong* private according to Def. 10 under an extended ODH assumption.*

Wide-strong* privacy of IBIHOP can be shown under an extended ODH assumption featuring the following oracles:

- $\mathcal{O}_1 = h$ (a handle h is returned to the value $e_h \in_R \mathbb{Z}_\ell$)
- $\mathcal{O}_2(Z, h) =$ (only to be invoked once per h)
  - $\mathcal{O}_{2a}$: $\texttt{xcoord}(bZ) \oplus e_h$ for $Z \neq \pm A$
  - $\mathcal{O}_{2b}$: $\texttt{xcoord}(C) \oplus e_h$ for $Z = \pm A$
- $\mathcal{O}_3(Z, h) = e_h^{-1} Z$
- $\mathcal{O}_4(z_1, z_2, Z) =$ (can only be invoked once)
  - $(\texttt{xcoord}(C) \oplus z_1)z_2 + a$, if $Z = (\texttt{xcoord}(C) \oplus z_1)^{-1}P$
  - $\perp$, otherwise

A similar privacy proof as for Theorem 7 can be used.

Note that $\mathcal{O}_4$ can only be called once during the ODH game, to prevent the value a from being leaked.

5.8 Private Threshold Things That Think

To protect the (location) privacy of the end-user, we need to make sure that all communication, and not merely the identification process, between his personal devices does not leak information about the identities of the personal devices. The provided partial decryptions or partial signatures (see Sect. 3.4) leak information, due to the homomorphic properties, about the originating device. To keep this information private, one can construct a private channel between the originating device and the combining device.[13] Furthermore, private channels are required at each instance of resharing (see Sect. 4.2). A mutual authentication protocol can be used to set up these private channels. Since we want to authenticate devices to devices, the support for multiple readers in our new model makes it applicable.

[13]Recall that a combining device collects all partial decryptions or partial signatures and computes the full decryption or full signature.

The proposed private mutual authentication protocol of the previous section, still leaks the identity of one of communicating devices. However, keep in mind that this protocol was designed for the general setting in which devices have no prior knowledge about the communication partners. Given the rather limited number of personal devices (and the even fewer number of possible combing devices among those), the approach of Paise-Vaudenay [124] is better suited. We need to append our private RFID authentication protocol with one extra message that will authenticate the reader directly to the tag.

5.9 Conclusion

In this chapter RFID privacy was approached from both the modelling and protocol point of view. Several RFID privacy models were critically examined with respect to their assumptions, practical usability and other issues that arise when applying their privacy definition to concrete protocols. Some models are based on unrealistic assumptions, others are impractical to apply. The privacy model of Hermans *et al.* [84], based on the classic notion of indistinguishability, combines the benefits of previous models while avoiding their identified drawbacks. Since this privacy model is based on an indistinguishability game, one can rely on a wide range of existing proof techniques, making the model quite straightforward to use in practice. We extended this model to allow for a more general setting where a tag can privately authenticate to multiple (independent) readers. Our extended model also incorporates the creation of insider tags, in order to also capture insider attacks. Finally we extended the model with a definition for mutual authentication.

From the protocol side, we examined several protocols towards their security and privacy properties. We proposed a new wide-strong private zero-knowledge RFID identification protocol. Previously, this notion was only achieved by protocols making use of an IND-CCA2 encryption scheme. Security and privacy of our protocol and all its optimised variant are proven in the standard model. Our protocol is the most efficient in its kind and can be implemented on RFID tags, using only Elliptic Curve Cryptography. This allows for a compact hardware design and requires minimal computational effort from the tag, namely two scalar-EC point multiplications. As an additional benefit, our protocols do not require any shared secrets between readers and tags. This makes these protocols very suitable for our extended setting with multiple readers. We also proposed the first efficient mutual authentication protocol that achieves wide-strong privacy.

Chapter 6

Conclusion

6.1 Conclusion

The main contribution of this thesis is the design of a security architecture that enables the use of threshold cryptography on personal devices. In this way, personal devices can be adequately secured, while steering away from the security-memorability trade-off. This solution takes into account that not all devices will be present constantly, i.e. the security system is resilient. An important parameter of this system is the threshold number. We discussed how to select the threshold number, with the total number of devices, to have an optimal balance between security and resilience.

We showed how to increase the resilience by making it possible to include small devices with limited, or even no (secure) storage capabilities. The only requirement for these devices is support for public-key functionality, more particularly Elliptic Curve Cryptography (ECC). Shares are stored in protected form and can be used implicitly, which makes it possible to store these externally. These protected shares can be generated in a distributed manner when at least the threshold number of devices is present. This means that not all devices need to participate in the group setup. In this way, the computational load for the resource constraint devices is reduced.

To make threshold cryptography on personal devices more practical, the end-user needs to be able to configure and update his set of personal devices. This is a necessary condition for the deployment of threshold cryptography on a set of personal devices over a longer time, e.g., a couple of years. More specifically, to ensure long-term security and to allow to recover from devices that are lost, stolen or forgotten, there should be a mechanism to replace devices or simply remove unused devices and add new devices. Although the underlying mechanism of resharing has already been studied, the question of how the end-user authorises these changes to his set of personal devices was yet unsolved. We designed a secure and user-friendly protocol that allows the consumer to authorise resharing and hence be in full control of his overall security system. We took the opportunity to create a graphical user interface for the protocol and conducted a preliminary usability study on it. How to deal with devices that are absent at the time of resharing was also discussed.

To protect the consumers' privacy, this thesis studied how to authenticate devices privately over an open channel. The exchanged messages, resulting from authentication protocol instances, do not allow any third party to track the end-user. We approached this challenge from the perspective of the smallest devices, namely RFID tags. Several RFID privacy models were critically examined and one was selected to extend. This extension allows to model private authentication to multiple readers, which is necessary for device to device authentication. It also models insider tags and provides modelling for mutual authentication. We proposed a new wide-strong private RFID identification protocol, the first of its kind that is based on zero-knowledge. Furthermore, our protocol can be implemented with a compact hardware design (only using an ECC coprocessor) and requires minimal computational effort from the tags. As an additional benefit, our protocols do not require any shared secrets between readers and tags. This makes these protocols very suitable for our extended setting with multiple readers. We also proposed the first efficient mutual authentication protocol that achieves wide-strong privacy.

6.2 Directions for Future Research

6.2.1 General Secret Sharing

In this thesis the focus was mainly on a threshold secret sharing scheme, in which all personal devices are treated equally. This is in shear contrast with reality where personal devices are quite different. The justification to opt for a threshold sharing scheme, despite the big differences among personal devices, was the following. One might be inclined to give a more important role to the more powerful personal devices since these usually come with better protection mechanisms against all sorts of attacks (e.g., physical attacks, side channel attacks). Yet, exactly these devices are of highest interest to adversaries. The more powerful devices are more expensive, i.e. have a higher direct value, and possibly contain (confidential) information of interest. This raises the question if it is not possible to come up with a more general secret scheme providing the flexibility to differentiate between devices.

Any monotone access structure can be transformed into a secret sharing scheme [90]. General models of secret sharing are given in [150, 157]. Specific relevant examples include weighted secret sharing [16, 118], hierarchical secret sharing [158] and multilevel secret sharing [122]. How to deploy these secret sharing schemes on things that think and comparing these with the case of threshold secret sharing is an interesting research question.

6.2.2 Usability

Threshold things that think has been proposed as another means of authenticating the end-user towards his personal devices. However, the general usability of threshold things that think versus other authentication mechanisms needs to be evaluated. Can we get the end-user to understand how it works? Which objects would the end-user pick to be part of this scheme? How would they carry those smart objects? At each time the end-user wants to authenticate, how many and which personal devices carry? How practical is this solution over time?

When comparing different secret sharing schemes, usability is an important factor. For example with the weighted secret sharing: How will the user assign weights? And what impact will this have on security?

6.2.3 Context

By taking context into account, user authentication can be enhanced and a more fine grained logical access control becomes possible without requiring additional effort from the end user. The context can be deduced from sensory data and the patterns in this data. So far, context has been studied for the one device scenario. Siewiorek *et al.* [149] created a mobile phone that is aware of its context, called SenSay. Stajano [155] pointed out that data on PDAs should be accessible according to the context the user is in, the user has different 'hats' and can decide which one to wear. Seifert *et al.* [146] implemented and evaluated the usability of mobile phone with three spheres: home, work and closed. Depending on the sphere the phone is in, different data are accessible. Their implementation supports locations and actions to switch between these spheres. Several research questions arise. What is the best way to assess the context in which personal devices are in? How can one prevent an adversary from tricking devices into thinking that these are in a different context? Which impact will this have on the privacy of the end-user?

Bibliography

[1] Aarts, E., and Wichert, R. Ambient Intelligence. In *Technology Guide*, H.-J. Bullinger, Ed. Springer, 2009, pp. 244–249. pages 2

[2] Abdalla, M., Bellare, M., and Rogaway, P. DHIES: An encryption scheme based on the Diffie-Hellman Problem. *Micro 2020* (2001), 1–30. pages 10

[3] Abe, M., and Fehr, S. Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography. In *Advances in Cryptology – CRYPTO 2004* (2004), M. K. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*, Springer, pp. 317–334. pages 22, 49

[4] Adams, A., and Sasse, M. A. Users are not the enemy. *Communications of the ACM 42*, 12 (1999), 40–46. pages 4, 62

[5] Aerts, W. *Application Specificities of Array Antennas: Satellite Communication and Electromagnetic Side Channel Analysis.* PhD thesis, Katholieke Universiteit Leuven, 2009. pages iv

[6] Armknecht, F., Sadeghi, A.-R., Scafuro, A., Visconti, I., and Wachsmann, C. Impossibility Results for RFID Privacy Notions. In *Transactions on Computational Science XI*, M. Gavrilova, C. Tan, and E. Moreno, Eds., vol. 6480 of *Lecture Notes in Computer Science.* Springer, 2010, pp. 39–63. pages 87, 97

[7] Ashton, K. That 'Internet of Things' Thing. RFID Journal article 4986, June 2009. pages 1

[8] Asmuth, C., and Bloom, J. A Modular Approach to Key Safeguarding. *IEEE Transactions on Information Theory 29*, 2 (1983), 208–210. pages 18

[9] Avanzi, R., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., and Vercauteren, F. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, vol. 34 of *Discrete Mathematics And Its Applications.* CRC Press, 2005. pages 7

[10] Avoine, G., Dysli, E., and Oechslin, P. Reducing Time Complexity in RFID Systems. In *Selected Areas in Cryptography* (2005), B. Preneel and S. E. Tavares, Eds., vol. 3897 of *Lecture Notes in Computer Science*, Springer, pp. 291–306. pages 84

[11] Backes, M., Kate, A., and Patra, A. Computational Verifiable Secret Sharing Revisited. In *Advances in Cryptology – ASIACRYPT '11* (2011), D. H. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*, Springer, pp. 590–609. pages 20

[12] Bao, F., Deng, R. H., and Zhu, H. Variations of Diffie-Hellman Problem. In *Information and Communications Security – ICICS 2003* (2003), S. Qing, D. Gollmann, and J. Zhou, Eds., vol. 2836 of *Lecture Notes in Computer Science*, Springer, pp. 301–312. pages 10

[13] Barić, N., and Pfitzmann, B. Collision-Free Accumulators and Fail-Stop Signature Schemes without Trees. In *International Conference on Theory and Application of Cryptographic Techniques* (1997), EUROCRYPT'97, Springer, pp. 480–494. pages 11

[14] Batina, L., Guajardo, J., Kerins, T., Mentens, N., Tuyls, P., and Verbauwhede, I. An Elliptic Curve Processor Suitable For RFID-Tags. In *Benelux Workshop on Information and System Security – WISSec 2006* (2006), 14 pages. pages 108

[15] Beaver, D., and Haber, S. Cryptographic Protocols Provably Secure Against Dynamic Adversaries. In *Advances in Cryptology – EUROCRYPT '92* (1992), R. A. Rueppel, Ed., vol. 658 of *Lecture Notes in Computer Science*, Springer, pp. 307–323. pages 22

[16] Beimel, A., Tassa, T., and Weinreb, E. Characterizing Ideal Weighted Threshold Secret Sharing. In *Theory of Cryptography – TCC '05* (2005), J. Kilian, Ed., vol. 3378 of *Lecture Notes in Computer Science*, Springer, pp. 600–619. pages 121

[17] Bellare, M., Fischlin, M., Goldwasser, S., and Micali, S. Identification Protocols Secure against Reset Attacks. In *Advances in Cryptology – EUROCRYPT '01* (2001), B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, Springer, pp. 495–511. pages 82

[18] Bellare, M., Namprempre, C., Pointcheval, D., and Semanko, M. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology 16* (2003), 185–215. pages 9

[19] Bellare, M., and Palacio, A. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In *Advances in Cryptology - CRYPTO 2002* (2002), vol. 2442 of *Lecture Notes in Computer Science*, Springer, pp. 162–177. pages 101

[20] Billet, O., Etrog, J., and Gilbert, H. Lightweight Privacy Preserving Authentication for RFID Using a Stream Cipher. In *International Workshop on Fast Software Encryption – FSE 2010* (2010), S. Hong and T. Itawa, Eds., vol. 6147 of *Lecture Notes in Computer Science*, Springer, pp. 55–74. pages 83

[21] Blakley, G. Safeguarding Cryptographic Keys. In *AFIPS National Computer Conference* (1979), AFIPS Press, pp. 313–317. pages 17

[22] Bleichenbacher, D. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *Advances in Cryptology – CRYPTO '98* (1998), H. Krawczyk, Ed., vol. 1462 of *Lecture Notes in Computer Science*, Springer, pp. 1–12. pages 87

[23] Blom, R. Non-Public Key Distribution. In *Advances in Cryptology – CRYPTO '82* (1982), D. Chaum, R. L. Rivest, and A. T. Sherman, Eds., Plenum Press, pp. 231–236. pages 15

[24] Bogetoft, P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J. D., Nielsen, J. B., Nielsen, K., Pagter, J., Schwartzbach, M., and Toft, T. Secure Multiparty Computation Goes Live. R. Dingledine and P. Golle, Eds., vol. 5628 of *Lecture Notes in Computer Science*, Springer, pp. 325–343. pages 15

[25] Bohli, J.-M., and Pashalidis, A. Relations Among Privacy Notions. In *Financial Cryptography and Data Security – FC '09* (2009), R. Dingledine and P. Golle, Eds., vol. 5628 of *Lecture Notes in Computer Science*, Springer, pp. 362–380. pages 90, 95

[26] Bringer, J., Chabanne, H., Cohen, G. D., and Kindarji, B. Private Interrogation of Devices via Identification Codes. In *Progress in Cryptology – INDOCRYPT 2009* (2009), B. K. Roy and N. Sendrier, Eds., vol. 5922 of *Lecture Notes in Computer Science*, Springer, pp. 272–289. pages 112

[27] Bringer, J., Chabanne, H., and Icart, T. Cryptanalysis of EC-RAC, a RFID Identification Protocol. In *International Conference on Cryptology and Network Security – CANS 2008* (2008), vol. 5339 of *Lecture Notes in Computer Science*, Springer, pp. 149–161. pages 98, 110

[28] Bringer, J., Chabanne, H., and Icart, T. Efficient Zero-Knowledge Identification Schemes which respect Privacy. In *Symposium on Information, Computer, and Communications Security* (2009), ASIACCS '09, ACM, pp. 195–205. pages 86, 96, 99, 107, 108, 110

[29] Brown, D. R. L. Generic Groups, Collision Resistance, and ECDSA. *Designs, Codes and Cryptography 35*, 1 (2005), 119–152. pages 8, 41

[30] Brown, D. R. L., and Gjøsteen, K. A Security Analysis of the NIST SP 800-90 Elliptic Curve Random Number Generator. In *Advances in cryptology – CRYPTO '07* (2007), A. Menezes, Ed., vol. 4622 of *Lecture Notes in Computer Science*, Springer, pp. 466–481. pages 9

[31] Burmester, M., Le, T., and Medeiros, B. Provably secure ubiquitous systems: Universally composable RFID authentication protocols. In *International Conference on Security and Privacy in Communication Networks – SECURECOMM* (2006), IEEE Press. pages 84

[32] Cachin, C., Kursawe, K., Lysyanskaya, A., and Strobl, R. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM conference on Computer and Communications Security – CCS '02* (2002), ACM, pp. 88–97. pages 20

[33] Canard, S., Coisel, I., Etrog, J., and Girault, M. Privacy-Preserving RFID Systems: Model and Constructions. Cryptology ePrint Archive, Report 2010/405, 2010. `http://eprint.iacr.org/`. pages 83, 84, 86, 88, 100, 101, 107, 110

[34] Canetti, R. *Studies in Secure Multiparty Computation and Applications.* PhD thesis, The Weizmann Institute of Science, 1996. pages 20

[35] Canetti, R. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Foundations of Computer Science – FOCS 2001* (2001), IEEE Computer Society, pp. 136–145. pages 22

[36] Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., and Rabin, T. Adaptive Security for Threshold Cryptosystems. In *Advances in Cryptology – CRYPTO 1999* (1999), M. J. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, Springer, pp. 98–115. pages 22, 34, 35

[37] Canetti, R., Goldreich, O., Goldwasser, S., and Micali, S. Resettable zero-knowledge (extended abstract). In *Symposium on Theory of Computing – STOC '00* (2000), F. F. Yao and E. M. Luks, Eds., ACM, pp. 235–244. pages 82

[38] Castro, M., and Liskov, B. Practical Byzantine Fault Tolerance. In *Third Symposium on Operating Systems Design and Implementation* (New Orleans, USA, 1999). pages 61

[39] Chen, M., Gonzalez, S., Vasilakos, A., Cao, H., and Leung, V. Body Area Networks: A Survey. *Mobile Networks and Applications 16* (2011), 171–193. pages 1

[40] Chor, B., Goldwasser, S., Micali, S., and Awerbuch, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Symposium on Foundations of Computer Science – FOCS '85* (1985), IEEE Computer Society, pp. 383–395. pages 19

[41] Computer Security Institute. *Computer Crime and Security Survey 2010/2011.* 2011. pages 3

[42] Cramer, R., Damgård, I., and Nielsen, J. B. Multiparty Computation from Threshold Homomorphic Encryption. In *Advances in Cryptology – EUROCRYPT 2001* (2001), B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, Springer, pp. 280–299. pages 35

[43] Cramer, R., and Shoup, V. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology – CRYPTO 1998* (1998), H. Krawczyk, Ed., vol. 1462 of *Lecture Notes in Computer Science*, Springer, pp. 13–25. pages 28, 44, 46, 100, 110

[44] Crosby, S. A., Goldberg, I., Johnson, R., Song, D. X., and Wagner, D. A Cryptanalysis of the High-Bandwidth Digital Content Protection System. In *Digital Rights Management Workshop* (2001), T. Sander, Ed., vol. 2320 of *Lecture Notes in Computer Science*, Springer, pp. 192–200. pages 15

[45] Damgård, I., and Jurik, M. A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography – PKC '01* (2001), vol. 1992 of *Lecture Notes in Computer Science*, Springer, pp. 119–136. pages 58

[46] Damgård, I., and Østergaard, M. RFID Security: Tradeoffs between Security and Efficiency. Cryptology ePrint Archive, Report 2006/234, 2006. `http://eprint.iacr.org/`. pages 84

[47] D'Arco, P., Scafuro, A., and Visconti, I. Revisiting DoS Attacks and Privacy in RFID-Enabled Networks. In *Algorithmic Aspects of Wireless Sensor Networks – ALGOSENSORS '09* (2009), S. Dolev, Ed., vol. 5804 of *Lecture Notes in Computer Science*, Springer, pp. 76–87. pages 86

[48] D'Arco, P., Scafuro, A., and Visconti, I. Semi-Destructive Privacy in RFID systems. In *Workshop on RFID Security – RFIDSec'09* (2009). pages 86

[49] Dargie, W., and Poellabauer, C. *Fundamentals of Wireless Sensor Networks: Theory and Practice.* Wireless Communications and Mobile Computing. Wiley, 2010. pages 1

[50] Desmedt, Y. Some Recent Research Aspects of Threshold Cryptography. In *Information Security Workshop – ISW 1997* (1997), M. Mambo, E. Okamoto, and E. Davida, Eds., vol. 1196 of *Lecture Notes in Computer Science*, Springer, pp. 158–173. pages 4

[51] Desmedt, Y., Burmester, M., Safavi-Naini, R., and Wang, H. Threshold Things That Think (T4): Security Requirements to Cope with Theft of Handheld/Handless Internet Devices. In *Symposium on Requirements Engineering for Information Security* (2001). pages 15

[52] Desmedt, Y., and Jajodia, S. Redistributing Secret Shares to New Access Structures and its Applications. Tech. Rep. ISSE-TR-97-01, George Mason University, 1997. pages 54

[53] Dolev, D., Dwork, C., Waarts, O., and Yung, M. Perfectly Secure Message Transmission. *J. ACM 40* (January 1993), 17–47. pages 20

[54] Feldman, P. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *Foundations of Computer Science – FOCS 1987* (1987), IEEE Computer Society, pp. 427–437. pages 12, 19

[55] Fiat, A., and Shamir, A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology – CRYPTO '86* (1986), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, Springer, pp. 186–194. pages 60

[56] Fouque, P.-A., and Stern, J. One Round Threshold Discrete-Log Key Generation without Private Channels. In *Public Key Cryptography – PKC 2001* (2001), K. Kim, Ed., vol. 1992 of *Lecture Notes in Computer Science*, Springer, pp. 300–316. pages 23

[57] Frankel, Y., Gemmell, P., MacKenzie, P. D., and Yung, M. Optimal Resilience Proactive Public-Key Cryptosystems. In *Symposium on Foundations of Computer Science – FOCS '97* (1997), vol. 1294, IEEE, pp. 384–393. pages 54

[58] Frankel, Y., Gemmell, P., MacKenzie, P. D., and Yung, M. Proactive RSA. In *Advances in Cryptology – CRYPTO '97* (1997), B. S. Kaliski, Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer, pp. 440–454. pages 53

[59] Frankel, Y., MacKenzie, P. D., and Yung, M. Adaptively-Secure Distributed Public-Key Systems. In *Algorithms – ESA 1999* (1999), J. Nesetril, Ed., vol. 1643 of *Lecture Notes in Computer Science*, Springer, pp. 4–27. pages 22, 34

[60] Frankel, Y., MacKenzie, P. D., and Yung, M. Adaptively-Secure Optimal-Resilience Proactive RSA. In *Advances in Cryptology – ASIACRYPT '99* (1999), E. O. Kwok-Yan Lam and C. Xing, Eds., vol. 1716 of *Lecture Notes in Computer Science*, Springer, pp. 180–194. pages 53

[61] Frankel, Y., MacKenzie, P. D., and Yung, M. Adaptive Security for the Additive-Sharing Based Proactive RSA. In *Workshop on Practice and Theory in Public Key Cryptography - PKC* (2001), K. Kim, Ed., vol. 1992 of *Lecture Notes in Computer Science*, Springer, pp. 240–263. pages 53

[62] Fujisaki, E., and Okamoto, T. Statistical zero knowledge protocols to prove modular polynomial relations. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology* (1997), CRYPTO '97, Springer, pp. 16–30. pages 11

[63] Fujisaki, E., and Okamoto, T. A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications. In *Advances in Cryptology – EUROCRYPT 1998* (1998), K. Nyberg, Ed., vol. 1403 of *Lecture Notes in Computer Science*, Springer, pp. 32–46. pages 21

[64] Galbraith, S., Hess, F., and Vercauteren, F. Aspects of Pairing Inversion. *IEEE Transactions on Information Theory 54*, 12 (2008), 5719–5728. pages 11

[65] Galbraith, S., Paterson, K., and Smart, N. Pairings for Cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. `http://eprint.iacr.org/`. pages 8, 34

[66] Gamal, T. E. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology – CRYPTO 1984* (1984), G. R. Blakley and D. Chaum, Eds., vol. 196 of *Lecture Notes in Computer Science*, Springer, pp. 10–18. pages 28, 44

[67] Gehrmann, C., Mitchell, C., and Nyberg, K. Manual Authentication for Wireless Devices. *RSA Cryptobytes 7*, 1 (2004), 29–37. pages 61

[68] Gehrmann, C., Nyberg, K., and Mitchell, C. The personal CA–PKI for Personal Area Network. In *Information Society Technologies Mobile and Wireless Communications Summit* (2002), pp. 31–35. pages 61

[69] Gennaro, R., Jarecki, S., Krawczyk, H., and Rabin, T. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In *Advances in Cryptology – EUROCRYPT 1999* (1999), J. Stern, Ed., vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 295–310. pages 19, 22, 35

[70] Gennaro, R., Jarecki, S., Krawczyk, H., and Rabin, T. Secure Applications of Pedersen's Distributed Key Generation Protocol. In *Topics in Cryptology – CT-RSA 2003* (2003), M. Joye, Ed., vol. 2612 of *Lecture Notes in Computer Science*, Springer, pp. 373–390. pages 22

[71] Gennaro, R., Jarecki, S., Krawczyk, H., and Rabin, T. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. *Journal of Cryptology 20*, 1 (2007), 51–83. pages 21, 36, 43, 49, 54

[72] Gilbert, H., Robshaw, M. J., and Seurin, Y. HB#: Increasing the Security and Efficiency of HB+. In *Advances in Cryptology – EUROCRYPT 2008* (2008), vol. 4965 of *Lecture Notes in Computer Science*, Springer, pp. 361–378. pages 83

[73] Girault, M., Poupard, G., and Stern, J. On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. *J. Cryptology 19* (2006), 463–487. pages 108

[74] Goldreich, O. *Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, 2001. pages 104

[75] Google. Getting started with 2-step verification. `http://support.google.com/accounts/bin/static.py?hl=en&page=guide.cs&guide=1056283`. pages 4

[76] Goyal, V., and Sahai, A. Resettably Secure Computation. In *Advances in Cryptology – EUROCRYPT '09* (2009), A. Joux, Ed., vol. 5479 of *Lecture Notes in Computer Science*, Springer, pp. 54–71. pages 82

[77] Guajardo, J., Škorić, B., Tuyls, P., Kumar, S. S., Bel, T., Blom, A. H., and Schrijen, G.-J. Anti-Counterfeiting, Key Distribution, and Key Storage in an Ambient World via Physical Unclonable Functions. *Information Systems Frontiers 11*, 1 (2009), 19–41. pages 28

[78] Ha, J., Moon, S.-J., Zhou, J., and Ha, J. A New Formal Proof Model for RFID Location Privacy. In *European Symposium on Research in Computer Security – ESORICS '08* (2008), S. Jajodia and J. López, Eds., vol. Lecture Notes in Computer Science, Springer, pp. 267–281. pages 84, 90

[79] Hansmann, U. *Pervasive Computing: the Mobile World.* Springer, 2003. pages 2

[80] Hardekopf, B., Kwiat, K., and Upadhyaya, S. A Decentralized Voting Algorithm for Increasing Dependability. In *Systemic, Cybernetics and Informatics (SCI2001)* (2001). pages 61

[81] Heidarvand, S., and Villar, J. L. Public Verifiability from Pairings in Secret Sharing Schemes. In *Selected Areas in Cryptography – SAC 2008* (2009), R. Avanzi, L. Keliher, and F. Sica, Eds., vol. 5381 of *Lecture Notes in Computer Science*, Springer, pp. 294–308. pages 21, 34

[82] Hein, D., Wolkerstorfer, J., and Felber, N. ECC Is Ready for RFID — A Proof in Silicon. In *Selected Areas in Cryptography – SAC '08* (2008), R. Avanzi, L. Keliher, and F. Si, Eds., vol. 5381 of *Lecture Notes in Computer Science*, Springer, pp. 401–413. pages 83

[83] Herley, C., and van Oorschot, P. C. A Research Agenda Acknowledging the Persistence of Passwords. *IEEE Security & Privacy 10*, 1 (2012), 28–36. pages 4

[84] Hermans, J., Pashalidis, A., Vercauteren, F., and Preneel, B. A New RFID Privacy Model. In *ESORICS 2011* (2011), V. Atluri and C. Diaz, Eds., vol. 6879 of *Lecture Notes in Computer Science*, Spri, pp. 568–587. pages 82, 91, 100, 117

[85] Hermans, J., and Peeters, R. Private Yoking Proofs: Attacks, Models and new Provable Constructions. In *Workshop on RFID Security 2012* (Nijmegen,NL, 2012), J.-H. Hoepman and I. Verbauwhede, Eds., Lecture Notes in Computer Science, Springer, 14 pages. pages 6

[86] Herzberg, A., Jakobsson, M., Jarecki, S., Krawczyk, H., and Yung, M. Proactive Public Key and Signature Systems. In *ACM Conference on Computer and Communications Security* (1997), ACM, pp. 100–110. pages 53

[87] Herzberg, A., Jarecki, S., Krawczyk, H., and Yung, M. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *Advances in Cryptology – CRYPTO '95* (1995), D. Coppersmith, Ed., vol. 963 of *Lecture Notes in Computer Science*, Springer, pp. 339–352. pages 53

[88] Hutter, M., Schmidt, J.-M., and Plos, T. RFID and Its Vulnerability to Faults. In *Cryptographic Hardware and Embedded Systems – CHES '08* (2008), E. Oswald and P. Rohatgi, Eds., vol. 5154 of *Lecture Notes in Computer Science*, Springer, pp. 363–379. pages 82

[89] ISO 9241-11. *Guidance on Usability.* International Organization for Standardization, 1998. pages 71

[90] Itoh, M., Saito, A., and Nishizeki, T. Secret sharing scheme realizing general access structure. In *IEEE Globecom* (1987), pp. 99–102. pages 121

[91] Jakobsson, M., Sako, K., and Impagliazzo, R. Designated Verifier Proofs and Their Applications. In *Advances in Cryptology – EUROCRYPT 1996* (1996), U. Maurer, Ed., vol. 1070 of *Lecture Notes in Computer Science*, Springer, pp. 143–154. pages 98

[92] Jarecki, S., and Lysyanskaya, A. Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. In *Advances in Cryptology - EUROCRYPT 2000* (2000), B. Preneel, Ed., vol. 1807 of *Lecture Notes in Computer Science*, Springer, pp. 221–242. pages 22, 23

[93] Joint European Commission / EPoSS Expert Workshop. Internet of Things in 2020. Tech. rep., 2008. pages 1, 2

[94] Juels, A., and Weis, S. A. Authenticating Pervasive Devices with Human Protocols. In *Advances in Cryptology – CRYPTO 2005* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 293–308. pages 83

[95] Juels, A., and Weis, S. A. Defining Strong Privacy for RFID. In *International Conference on Pervasive Computing and Communications - Workshops (PerCom Workshops 2007)* (2007), IEEE Computer Society, pp. 342–347. pages 84, 89

[96] Juels, A., and Weis, S. A. Defining Strong Privacy for RFID. *ACM Trans. Inf. Syst. Secur. 13* (November 2009), 7:1–7:23. pages 89

[97] JURIK, M. J. *Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols.* PhD thesis, BRICS, Department of Computer Science, University of Aarhus, 2004. pages 58

[98] KAHN, J. M., KATZ, R. H., KATZ, Y. H., AND PISTER, K. S. J. Emerging Challenges: Mobile Networking for "Smart Dust". *Journal of Communications and Networks 2* (2000), 188–196. pages 1

[99] KASPER, T., OSWALD, D., AND PAAR, C. New Methods for Cost-Effective Side-Channel Attacks on Cryptographic RFIDs. In *Workshop on RFID Security – RFIDSec'09* (2009, 15 pages). pages 82

[100] KATE, A., AND GOLDBERG, I. Distributed Key Generation for the Internet. In *IEEE International Conference on Distributed Computing Systems* (2009), IEEE Computer Society, pp. 119–128. pages 23

[101] KRAWCZYK, H. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In *Advances in Cryptology - CRYPTO 2001* (2001), J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, Springer, pp. 310–331. pages 101

[102] LAGUILLAUMIE, F., AND VERGNAUD, D. Designated Verifier Signatures: Anonymity and Efficient Construction from Any Bilinear Map. In *Security in Communication Networks* (2004), C. Blundo and S. Cimato, Eds., vol. 3352 of *Lecture Notes in Computer Science*, Springer, pp. 105–119. pages 98

[103] LANDROCK, P., AND PEDERSEN, T. WYSIWYS? – What You See Is What You Sign? *Information Security Technical Report 3*, 2 (1998), 55 – 61. pages 64

[104] LAUR, S., ASOKAN, N., AND NYBERG, K. Efficient Mutual Data Authentication Using Manually Authenticated Strings. Cryptology ePrint Archive, Report 2005/424, 2005. pages 61

[105] LAUR, S., AND PASINI, S. SAS-Based Group Authentication and Key Agreement Protocols. In *11th International Workshop on Practice and Theory in Public-Key Cryptography – PKC '08* (2008), R. Cramer, Ed., vol. 4939 of *Lecture Notes in Computer Science*, Springer, pp. 197–213. pages 61, 67

[106] LEE, Y. K., BATINA, L., SAKIYAMA, K., AND VERBAUWHEDE, I. Elliptic Curve Based Security Processor for RFID. *IEEE Transactions on Computers 57*, 11 (2008), 1514–1527. pages 83

[107] Lee, Y. K., Batina, L., Singelée, D., and Verbauwhede, I. Low-Cost Untraceable Authentication Protocols for RFID. In *ACM conference on Wireless Network Security – WiSec '10* (2010), C. Nita-Rotaru and F. Stajano, Eds., ACM, pp. 55–64. pages 107

[108] Lewis, J. R. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction 7*, 1 (1995), 57–78. pages 71

[109] Liu, C. L. *Introduction to Combinatorial Mathematics.* McGraw-Hill, 1968. pages 17

[110] Lomb, B., and Güneysu, T. Decrypting HDCP-protected Video Streams Using Reconfigurable Hardware. In *ReConFig* (2011), IEEE Computer Society, pp. 249–254. pages 15

[111] Maes, R., Peeters, R., Van Herrewege, A., Wachsmann, C., Katzenbeisser, S., Sadeghi, A.-R., and Verbauwhede, I. Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-enabled RFIDs. In *International Conference on Financial Cryptography and Data Security – FC '12* (2012), Lecture Notes in Computer Science, Springer, 16 pages. pages 6

[112] Malone-Lee, J., and Smart, N. P. Modifications of ECDSA. In *Selected Areas in Cryptography – SAC 2002* (2002), K. Nyberg and H. M. Heys, Eds., vol. 2595 of *Lecture Notes in Computer Science*, Springer, pp. 1–12. pages 41

[113] Mangard, S., Oswald, E., and Popp, T. *Power analysis attacks - revealing the secrets of smart cards.* Springer, 2007. pages 82

[114] Mannan, M., and van Oorschot, P. C. Leveraging personal devices for stronger password authentication from untrusted computers. *Journal of Computer Security 19* (2011), 703–750. pages 4

[115] Mao, W. *Modern Cryptography: Theory and Practice.* Prentice Hall, 2003. pages 9

[116] Mignotte, M. How to Share a Secret. In *Proceedings of the Workshop on Cryptography* (1982), T. Beth, Ed., vol. 149 of *Lecture Notes in Computer Science*, pp. 371–375. pages 18

[117] Molich, R., and Nielsen, J. Improving a Human-Computer Dialogue. *Communications of the ACM 33*, 3 (1990), 338–348. pages 74

[118] Morillo, P., Padró, C., Sáez, G., and Villar, J. Weighted Threshold Secret Sharing Schemes. *Information Processing Letters 70*, 5 (1999), 211 – 216. pages 121

[119] Ng, C. Y., Susilo, W., Mu, Y., and Safavi-Naini, R. RFID Privacy Models Revisited. In *European Symposium on Research in Computer Security – ESORICS '08* (2008), S. Jajodia and J. López, Eds., vol. 5283 of *Lecture Notes in Computer Science*, Springer, pp. 251–266. pages 86, 87

[120] Ng, C. Y., Susilo, W., Mu, Y., and Safavi-Naini, R. New Privacy Results on Synchronized RFID Authentication Protocols against Tag Tracing. In *European Symposium on Research in Computer Security – ESORICS '09* (2009), M. Backes and P. Ning, Eds., vol. 5789 of *Lecture Notes in Computer Science*, Springer, pp. 321–336. pages 86

[121] Nguyen, L. H., and Roscoe, A. W. Efficient group authentication protocols based on human interaction. Cryptology ePrint Archive, Report 2009/150, 2009. `http://eprint.iacr.org/`. pages 61

[122] Padro, C., and Saez, G. Secret sharing schemes with bipartite access structure. *Information Theory, IEEE Transactions on 46*, 7 (nov 2000), 2596 –2604. pages 121

[123] Paillier, P. Public-Key Cryptosystems based on Composite Degree Residue Classes. In *Advances in Cryptology – EUROCRYPT '99* (1999), vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 223–238. pages 58

[124] Paise, R.-I., and Vaudenay, S. Mutual Authentication in RFID: Security and Privacy. In *ASIACCS'08* (2008), ACM Press, pp. 292–299. pages 86, 111, 117

[125] Pashalidis, A. Accessing Password-Protected Resources without the Password. In *Congress on Computer Science and Information Engineering – CSIE '09* (2009), M. Burgin, M. H. Chowdhury, C. H. Ham, S. A. Ludwig, W. Su, and S. Yenduri, Eds., IEEE Computer Society, pp. 66–70. pages 4

[126] Pedersen, T. P. A Threshold Cryptosystem without a Trusted Party. In *Advances in Cryptology - EUROCRYPT '91* (1991), vol. 547 of *Lecture Notes in Computer Science*, Springer, pp. 522–526. pages 22

[127] Pedersen, T. P. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology - CRYPTO '91* (1992), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer, pp. 129–140. pages 12, 19, 20, 31

[128] Peeters, R., Kohlweiss, M., and Preneel, B. Threshold Things That Think: Authorisation for Resharing. In *Proceedings of iNetSec 2009 – Open Research Problems in Network Security* (2009), J. Camenisch and D. Kesdogan, Eds., vol. 309 of *IFIP Advances in Information and Communication Technology*, pp. 111–124. pages 5

[129] Peeters, R., Kohlweiss, M., Preneel, B., and Sulmon, N. Threshold Things That Think: Usable Authorisation for Resharing. In *Symposium on Usable Privacy and Security - SOUPS 2009* (2009), L. F. Cranor, Ed., ACM, p. 2. pages 5

[130] Peeters, R., Simoens, K., De Cock, D., and Preneel, B. Cross-Context Delegation through Identity Federation. In *Proceedings of the Special Interest Group on Biometrics and Electronic Signatures* (2008), A. Brömme, C. Busch, and D. Hühnlein, Eds., vol. P-137 of *Lecture Notes in Informatics (LNI)*, Bonner Köllen Verlag, pp. 79–92. pages 6

[131] Peeters, R., Singelée, D., and Preneel, B. Threshold-Based Distance Bounding. In *International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use 2010* (2010), ACM, pp. 24–30. pages 6

[132] Peeters, R., Singelée, D., and Preneel, B. Threshold-Based Location-Aware Access Control. *International Journal of Handheld Computing Research 2*, 3 (2011), 22–37. pages 6

[133] Peeters, R., Singelée, D., and Preneel, B. Towards More Secure and Reliable Access Control. *IEEE Pervasive Computing 11*, 3 (July–September 2012), 76–83. pages 6

[134] Plos, T. Evaluation of the Detached Power Supply as Side-Channel Analysis Countermeasure for Passive UHF RFID Tags. In *Topics in Cryptology – CT-RSA '09* (2009), M. Fischlin, Ed., vol. 5473 of *Lecture Notes in Computer Science*, Springer, pp. 444–458. pages 82

[135] Pointcheval, D., and Stern, J. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology 13*, 3 (2000), 361–396. pages 49

[136] Rabin, T. A Simplified Approach to Threshold and Proactive RSA. In *Advances in Cryptology – CRYPTO '98* (1998), H. Krawczyk, Ed., vol. 1462 of *Lecture Notes in Computer Science*, Springer, pp. 89–104. pages 53

[137] Rasmussen, K. B., and Čapkun, S. Realization of RF Distance Bounding. In *USENIX Security Symposium '10* (2010), USENIX, pp. 389–402. pages 96

[138] SADEGHI, A.-R., VISCONTI, I., AND WACHSMANN, C. Anonymizer-Enabled Security and Privacy for RFID. In *Cryptology and Network Security – CANS '09* (2009), J. A. Garay, A. Miyaji, and A. Otsuka, Eds., vol. 5888 of *Lecture Notes in Computer Science*, Springer, pp. 134–153. pages 86

[139] SADEGHI, A.-R., VISCONTI, I., AND WACHSMANN, C. Efficient RFID security and privacy with anonymizers. In *Workshop on RFID Security – RFIDSec'09* (2009). pages 86

[140] SALTZER, J., AND SCHROEDER, M. The protection of Information in Computer Systems. *Proceedings of the IEEE 63*, 9 (1975), 1278–1308. pages 62

[141] SAXENA, N., UDDIN, M. B., AND VORIS, J. Universal Device Pairing using an Auxiliary Device. In *Symposium on Usable Privacy and Security – SOUPS 2008* (2008), ACM, pp. 56–67. pages 61

[142] SCHNORR, C.-P. Efficient Identification and Signatures for Smart Cards. In *Advances in Cryptology – CRYPTO 1989* (1989), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer, pp. 239–252. pages 28, 44, 49

[143] SCHNORR, C.-P. Efficient Signature Generation by Smart Cards. *Journal of Cryptology 4*, 3 (1991), 161–174. pages 101, 113

[144] SCHOENMAKERS, B. A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting. In *Advances in Cryptology – CRYPTO 1999* (1999), M. J. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, Springer, pp. 148–164. pages 21, 29

[145] SCHULTZ, D., LISKOV, B., AND LISKOV, M. MPSS: Mobile Proactive Secret Sharing. *ACM Trans. Inf. Syst. Secur. 13* (December 2010), 34:1–34:32. pages 20

[146] SEIFERT, J., DE LUCA, A., CONRADI, B., AND HUSSMANN, H. TreasurePhone: Context-Sensitive User Data Protection on Mobile Phones. In *Pervasive 2010* (2010), P. Floréen, A. Krüger, and M. Spasojevic, Eds., vol. 6030 of *Lecture Notes of Computer Science*, Springer, pp. 130–137. pages 121

[147] SHA-3 ZOO. Overview of all candidates for the current SHA-3 hash competition organized by NIST. `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`. pages 102, 109

[148] SHAMIR, A. How to Share a Secret. *Communications of the ACM 22*, 11 (1979), 612–613. pages 17

[149] Siewiorek, D., Smailagic, A., Furukawa, J., Krause, A., Moraveji, N., Reiger, K., Shaffer, J., and Wong, F. L. SenSay: A Context-Aware Mobile Phone. In *International Symposium on Wearable Computers – ISWC '03* (2003), IEEE Computer Society, pp. 248–249. pages 121

[150] Simmons, G. J. How to (Really) Share a Secret. In *Advances in Cryptology - CRYPTO '88* (1988), S. Goldwasser, Ed., vol. 403 of *Lecture Notes in Computer Science*, Springer, pp. 390–448. pages 121

[151] Simoens, K., Peeters, R., and Preneel, B. Increased Resilience in Threshold Cryptography: Sharing a Secret with Devices That Cannot Store Shares. In *Pairing-Based Cryptography - Pairing 2010* (2010), M. Joye, A. Miyaji, and A. Otsuka, Eds., vol. 6487 of *Lecture Notes in Computer Science*, Springer, pp. 116–135. pages 5

[152] Smart, N. P., and Vercauteren, F. On Computable Isomorphisms in Efficient Asymmetric Pairing-Based Systems. *Discrete Applied Mathematics 155*, 4 (2007), 538–547. pages 9

[153] Sohraby, K., Minoli, D., and Znati, T. *Wireless Sensor Networks: Technology, Protocols and Applications.* Wiley-Interscience, 2007. pages 1

[154] Stadler, M. Publicly Verifiable Secret Sharing. In *Advances in Cryptology – EUROCRYPT 1996* (1996), U. M. Maurer, Ed., vol. 1070 of *Lecture Notes in Computer Science*, Springer, pp. 190–199. pages 21

[155] Stajano, F. One User, Many Hats; and, Sometimes, No Hat: Towards a Secure yet Usable PDA. In *International conference on Security Protocols* (2004), J. A. M. Bruce Christianson, Bruno Crispo and M. Roe, Eds., vol. 3957 of *Lecture Notes in Computer Science*, Springer, pp. 51–64. pages 121

[156] Stajano, F. Pico: No More Passwords! In *Security Protocols Workshop* (2011), B. Christianson, B. Crispo, J. A. Malcolm, and F. Stajano, Eds., vol. 7114 of *Lecture Notes in Computer Science*, Springer, pp. 49–81. pages 4

[157] Stinson, D. R. An Explication of Secret Sharing Schemes. *Des. Codes Cryptography 2*, 4 (1992), 357–390. pages 121

[158] Tassa, T. Hierarchical Threshold Secret Sharing. *J. Cryptology 20*, 2 (2007), 237–264. pages 121

[159] Ullah, S., Higgins, H., Braem, B., Latre, B., Blondia, C., Moerman, I., Saleem, S., Rahman, Z., and Kwak, K. A

Comprehensive Survey of Wireless Body Area Networks. *Journal of Medical Systems* (2010), 1–30. 10.1007/s10916-010-9571-3. pages 1

[160] Van Deursen, T., and Radomirović, S. Insider attacks and privacy of RFID protocols. In *EuroPKI 2011* (2011), Lecture Notes in Computer Science, Springer, pp. 65–80. To appear. pages 92, 113

[161] Van Le, T., Burmester, M., and de Medeiros, B. Universally composable and forward-secure RFID authentication and authenticated key exchange. In *ACM Symposium on Information, Computer and Communications Security – ASIACSS '07* (New York, NY, USA, 2007), ACM, pp. 242–252. pages 84

[162] Vaudenay, S. On Privacy Models for RFID. In *Advances in Cryptology - ASIACRYPT 2007* (2007), K. Kurosawa, Ed., vol. 4833 of *Lecture Notes in Computer Science*, Springer, pp. 68–87. pages 83, 84, 85, 86, 90, 100, 110, 111, 113

[163] Vaudenay, S. Invited talk at RFIDSec 2010, 2010. pages 87

[164] Wang, S., and Safavi-Naini, R. New Results on Unconditionally Secure Multi-receiver Manual Authentication. In *2nd International Conference on Information Theoretic Security - ICITS 2007* (2007), Lecture Notes in Computer Science, Springer. pages 61

[165] Warneke, B., Last, M., Liebowitz, B., and Pister, K. Smart Dust: Communicating with a Cubic-Millimeter Computer. *Computer 34*, 1 (January 2001), 44 –51. pages 1

[166] Weis, S. A., Sarma, S. E., Rivest, R. L., and Engels, D. W. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *Security in Pervasive Computing – SPC '03* (2003), D. Hutter, G. Müller, W. Stephan, and M. Ullmann, Eds., vol. 2802 of *Lecture Notes in Computer Science*, Springer, pp. 201–212. pages 81

[167] Weiser, M. The Computer for the 21st Century. *Scientific American 3*, 3 (1991), 3–11. pages 2

[168] Weiser, M., Gold, R., and Brown, J. S. The Origins of Ubiquitous Computing Research at PARC in the Late 1980s. *IBM Systems Journal 38*, 4 (1999), 693–696. pages 2

[169] Wenger, E., and Hutter, M. A Hardware Processor Supporting Elliptic Curve Cryptography for Less Than 9 kGEs. In *Smart Card Research and Advanced Applications – CARDIS 2011* (2011), E. Prouff, Ed., vol. 7079 of *Lecture Notes in Computer Science*, Springer, pp. 182–199. pages 107

[170] Whitten, A., and Tygar, J. D. Usability of security: A case study. Tech. rep., CMU-CS-98-155, 1998. pages 62

[171] Whitten, A., and Tygar, J. D. Why Johnny Can't Encrypt: a Usability Evaluation of PGP 5.0. In *8th USENIX Security Symposium* (1999), USENIX Association, pp. 14–14. pages 74

[172] Wong, T. M., Wang, C., and Wing, J. M. Verifiable Secret Redistribution for Threshold Sharing Schemes. Tech. Rep. CMU-CS-02-114, Carnegie Mellon University, 2002. pages 54, 55, 57

[173] Yao, A. C.-C. Theory and Applications of Trapdoor Functions (Extended Abstract). In *Foundations of Computer Science – FOCS 1982* (1982), IEEE Computer Society, pp. 80–91. pages 104

[174] Yee, K.-P. User Interaction Design for Secure Systems. In *International Conference on Information and Communications Security – ICICS 2002* (2002), R. H. Deng, S. Qing, F. Bao, and J. Zhou, Eds., vol. 2513 of *Lecture Notes in Computer Science*, Springer, pp. 278–290. pages 74, 75

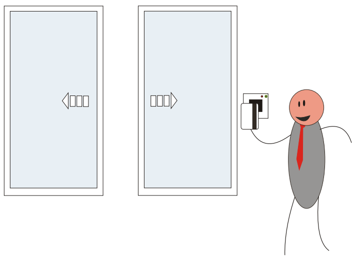
[175] Zhou, L., Schneider, F. B., and Van Renesse, R. APSS: Proactive Secret Sharing in Asynchronous Systems. *ACM Trans. Inf. Syst. Secur. 8* (August 2005), 259–286. pages 20

[176] Zuo, Y. Secure and Private Search Protocols for RFID Systems. *Information Systems Frontiers 12*, 5 (2010), 507–519. pages 112

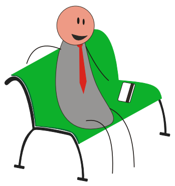
Appendix A

Treshold Things That Think in Pictures

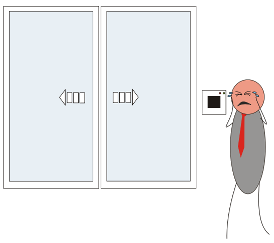
These cartoons were developped to introduce the concept of Threshold Things That Think to the general public in the context of the conducted usubility study (Sect. 4.4.3). Figures A.1 and A.2 focus on the scenario of access control to a building. Figures A.3 and A.4 consider data protection on a laptop.



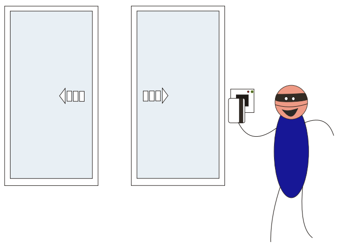
(a) You have an access card and with it you can access restricted parts of the company you are working in.



(b) If you leave your access card on the bench in the park where you had lunch ...

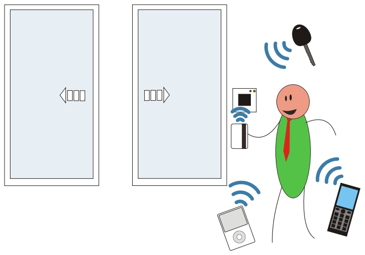


(c) ... you can no longer access restricted parts and not get to your working place.

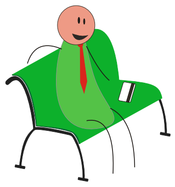


(d) However, someone finding your access card is granted access.

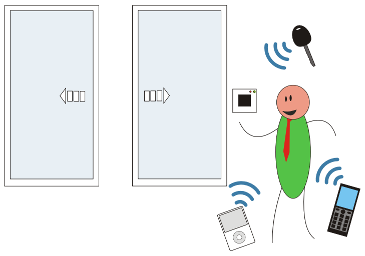
Figure A.1: Access Control Nowadays.



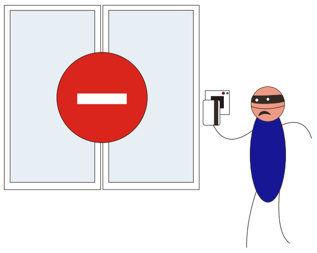
(a) Multiple personal devices together allow you access to restricted parts of the company you are working in.



(b) If you leave your access card on the bench in the park where you had lunch ...



(c) ... you can still access restricted parts and get to your working place.

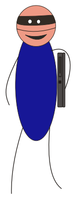


(d) Someone finding only your access card is not granted access.

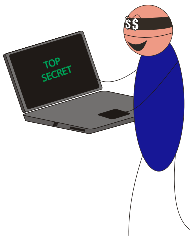
Figure A.2: Access Control with Threshold Things That Think Technology.

(a) Your laptop contains sensitive data.

(b) A burglar breaks in after midnight ...



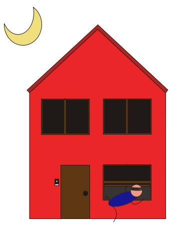
(c) ... and steals your laptop.



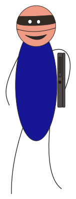
(d) Not only he has your laptop, he also has access to your sensitive data.

Figure A.3: Data Protection Nowadays.

(a) Your laptop contains sensitive data. If you want to view/work on this data you need multiple personal devices together.



(b) A burglar breaks in after midnight ...



(c) ... and steals your laptop

(d) He has your laptop, but he has no access to your sensitive data.

Figure A.4: Data Protection with Threshold Things That Think Technology.