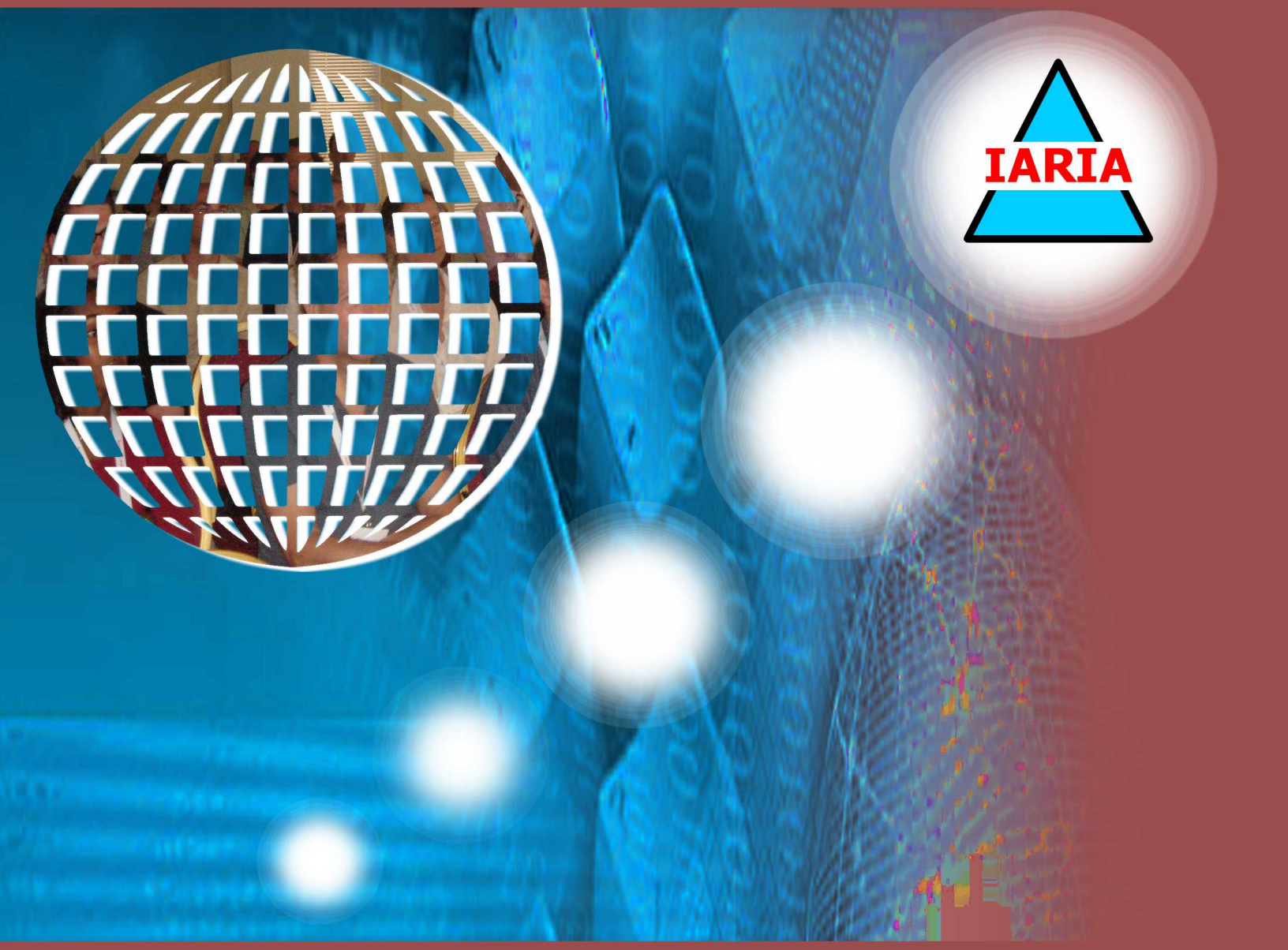
International Journal on Advances in Security



2011 vol. 4 nr. 1&2

The *International Journal on Advances in Security* is published by IARIA.
ISSN: 1942-2636
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Security
Volume 4, Number 1 & 2, 2011

- Bjarne E. Helvik, The Norwegian University of Science and Technology (NTNU) – Trondheim, Norway
- Aljosa Pasic, ATOS Origin, Spain
- Vladimir Stantchev, Berlin Institute of Technology, Germany
- Michiaki Tatsubori, IBM Research - Tokyo Research Laboratory, Japan
- Ian Troxel, SEAKR Engineering, Inc., USA
- Hans P. Zima, Jet Propulsion Laboratory/California Institute of Technology - Pasadena, USA // University of Vienna, Austria

Security in Internet

- Evangelos Kranakis, Carleton University, Canada
- Clement Leung, Victoria University - Melbourne, Australia
- Sjouke Mauw, University of Luxembourg, Luxembourg
- Yong Man Ro, Information and Communication University - Daejon, South Korea

International Journal on Advances in Security
Volume 4, Numbers 1 & 2, 2011

CONTENTS

Role of User Profile in Cloud-based Collaboration Services for Innovation

Zahid Iqbal, Josef Noll and Sarfraz Alam

University Graduate Center (UNIK)
Kjeller, Norway
{zahid, josef, sarfraz}@unik.no

***Abstract*—Enterprises are evolving their businesses from silo-based knowledge to collaborative-based knowledge by promoting open innovation through collaboration in to their technology infrastructure. Despite of being a prevailing trend, enterprises are not quite willing to embrace the collaboration into their working environment. This unwillingness is due to number of technical obstacles including user profiling, balancing of open and close collaboration and trust establishment. Therefore, the paper tries to address these impediments by contemplating collaborative enterprise computing approach that creates the network of enterprises for enabling the active, automated and trusted inter-enterprise collaboration. We propose a privacy-enhanced innovation framework that eases off the innovation process in an open and control manner. The framework does not only allow enterprise employees to create a user profile but also encourage them to initiate innovation activity by registering their novel ideas, which later can be realized in the form of business opportunity. We select an "innovation stock exchange" case study in order to apply the proposed approach. Furthermore, we intend to implement the framework in the form of cloud services that are interoperable with any enterprise collaboration platform.**

***Keywords*-Cloud Computing, Collaboration, Enterprise, Innovation, Privacy, Social Computing, Trust, User Profile**

I. INTRODUCTION

While open innovation is a prevailing trend that could spur new business opportunities, but enterprises are reluctant to adopt and invest in open innovation initiatives due to the risk factors associated with it. Most of the enterprises consider such initiatives as potential channel of loss of knowledge, control and core competencies, which in turns could negatively impact the enterprise long term innovation life cycle. However, success of consumer based social computing compels the enterprises to tend towards collaborative knowledge environment where the inter-enterprise boundary line is becoming indistinct by braking the silos. Today, collaboration is invertible for an enterprise in order to meet the rapid and dynamic demands of the businesses. Recently, HP Labs started the initiative to conduct open innovation by establishing the Innovative Research Program (IRP) between universities, enterprises and governments [1]. This fosters the collaboration among different participants in the form of sharing new ideas, enabling people to work across enterprise boundaries. It creates opportunities for capturing relevant knowledge, expertise so that innovative products and services could be introduced to the market. Initial indicators from HP Labs show that active collaboration helps in augmenting and accelerating knowledge creation and technology transfer, and the result of IRP is 179 papers and 34 HP patent disclosures in just three years [1].

Bringing innovation through collaboration is relied on enabling tool sets that allow enterprises to collaborate beyond their perimeter in a trusted open environment. Mostly, enterprises use email as an enabling tool for collaboration, where the new idea owner invites others through email to provide feedback and collaborate on its innovative idea that could be realized to a business opportunity by involving some external partners. This approach limits the collaboration space to the personal contacts of the idea owner within and outside of the enterprise. It is highly probable that the most suitable partners will be missed since they are not in direct contact of the idea owner.

In order to foster an active collaborative environment, enterprises can benefit from social computing platforms because these platforms promote sharing and openness within communities. These platforms offer different functionalities such as sharing of knowledge and idea, displaying recent activities of people, showing contacts and skills of people, and providing a list of colleagues and friends from social network sites. Enterprises are also considering social computing platforms to communicate with their customers and inform them about new services and releases. This does not only bring value and uptake for their business in the form of enhanced productivity and revenue, but also provides customers with the benefits of receiving services that are pertinent to their preferences. A survey report from McKinsey Global indicates that enterprises have gained high-business benefits by integrating social computing platforms in their working environment [2]. A range of studies [3][4][5] pointed out the significant of social computing, but only few [6][7] addressed the challenges and opportunities of social computing in an enterprise environment. Though enterprises can possess significant benefits from consumer based social computing platforms such as Facebook, but stringent security and privacy requirement of an enterprise does not foster use of such platform for collaboration. This amplifies the need of equipping innovation platform with corporate social computing platforms so that stringent enterprise security and privacy requirements can be enforced. Current state-of-art collaborative platforms, such as Microsoft's SharePoint and IBM's Lotus, only support

intra-enterprise collaboration and deficit in providing collaboration beyond the enterprise perimeter. However, extending collaboration space beyond the enterprise perimeter brings some new research challenges, which needs to be addressed. First one is the user profiling, which is the foundation of the multitude of functionalities within any collaborative platform. User profile is the core for automated collaboration for any enterprise. It allows to perform different numbers of interesting scenarios, including people search based on their expertise, target content push towards users to accelerate the knowledge sharing. User profiles are essential for an enterprise, containing information about the people working in an organization and helping to obtain appropriate information about people's skills, education, and contacts. As user profiles are not linked, it is not possible to reuse existing user profile on any other site. Even state-of-art social computing platforms do not facilitate any mechanism for linking user profiles to objects such as people, device, data and sensors. Second one is balancing of open and close collaboration for innovation in a user-centric way by specifying collaboration criterion. We envisage that the future lies in an innovation process that are under perceived control of each innovation initiative owner. In many cases, enterprises do not willing to share knowledge with certain competitors in order to serve the increasing demands of shorter innovation life cycle, and create successful products faster than their competitors to protect their business. This leads to third challenge of establishing trust and forming trusted virtual enterprise, connecting a number of autonomous enterprises that collaborate to achieve either a common business goal or to form a virtual market place.

In this paper, we propose a privacy-enhanced collaborative framework for an enterprise that operates in an open-controlled environment. The proposed approach aims at easy and automated collaborative process within and beyond enterprises perimeter for bringing innovation by sharing knowledge and technology transfer. The idea is to form an innovation cloud by leveraging the cloud computing that facilitate and speed up the innovation life cycle. This provides an entry point especially to SMEs and making them active part of the innovation process. The proposed framework comprises of three-components: Access Manager (AM), Collaboration Ensembler (CE) and Collaboration Criterion Manager (CCM). The framework supports user profile management and enables enterprises employees to register their profiles. It further allows them to initiate any novel idea or business opportunity. The user profiles and ideas are backed up with profile and idea database whose schema is structured though semantic enhanced models which are specified in the form of user profile ontology, trust ontology and idea ontology.

The rest of the paper is structured as follows. Section *II* discusses related work. Section *III* outlines the collaboration enterprise computing approach. Section *IV* provides details of the case study where proposed approach is employed. Section *V* discusses how automated collaboration can be enabled in an enterprise environment. Section *VI* presents the details of proposed framework. Section *VII* provides the road map to realized proposed framework in the form of cloud services and integrate those services with existing enterprise collaboration platforms. We conclude the paper in Section *VIII*.

II. Related Work

This section presents the overview of prior work in the area of user profiling, and cloud-based collaboration computing.

A. User Profiling

Social Network sites such as Facebook, Orkut, and MySpace allow people to share their interests, social information and contents among their friends or group of friends. It had been seen in the very beginning that the information which is stored on social network sites are not under the user control. All the information is owned or controlled by database owner. The profiles which contains user's personal information and attributes are typically cannot be exported in machine processable formats. The lack of machine understandability is a big hindrance in data portability and transformation between systems.The aforementioned drawbacks can be resolved with the advent of semantic technologies [8], more specifically the Friend of a Friend (FOAF) project [9] which was initiated by Dan Brickley and Libby Miller in 2000. Friend of a Friend (FOAF) contains RDF vocabulary for expressing user personal information (i.e., homepage, interest, friends, etc.) to create FOAF profiles which are shared among people in a distributed manner. FOAF profiles are posted on personal web site of the user and linked from the user's homepage. FOAF profiles are static in nature and contains only one term "knows" for describing social relationships. FOAF profiles also contain only one term "interest" for describing user's interest in a specific topic. FOAF profiles does not provide any vocabulary for capturing user's context. Hence, FOAF profiles are only for describing and linking people and things but not best suited to address user profile for personalized and context-aware service delivery.

Gosh et. al. present technique of creation and discovery of user profile [10]. The discovery of appropriate user profile for specific service is addressed by considering the user sparse information and context-awareness impact while accessing services.Dynamically construction of user profile is done by Profile Mediator and Constructor that receives desired user profile information for requesting services. The user profile ontology is defined in OWL by reusing FOAF vocabulary. Thus, it inherits the same aforementioned FOAF limitations.

The 3rd Generation Partnership Project (3GPP) is a major standardization body dealing with future 3G networks and services. The Generic User Profile (GUP) spcecification [11] is one of the 3GPP initiative to provide personalized services delivery within the operators domain. The GUP aggregates user related information such as user description, user services, and user devices to provide personalized service delivery in a standardized manner. GUP defines a global schema of the user profile in XML. Though, GUP is a well-known specification for user profiling but it lacks the enrichment of user profile and since it is based on XML so it cannot provide any intended

meaning to associated data and only constrain the structure of GUP profile.

Stann et. al. presents user profile ontology [14] which is inspired by the SPICE project [15]. A dynamic, situation aware user profile ontology is represented which enables real time situation awareness of the user and to express the social network related preferences in situational sub-profiles.The preferences are only limited to how a person's friend or category of friend can reach him in a specific situation and how Services (vibrate, ring, voice message) can inform or notify him from a mobile phone.

The General User Model Ontology (GUMO) [12] is a notorious user profile ontology, represented using OWL. The GUMO inherits the UserML [13] approach where user profile is divided into triples. It contains some basic and useful information about user's characteristics, emotional state and some facts about user's personality. However, GUMO is quasi-static model where applications can retrieve and add information into the profile.

B. Collaboration and Cloud Computing

Rapid adoption of social computing not only brings a new collaborative and innovation business opportunity for enterprises but also leads to the issue of corporate privacy when the collaboration is formed within the trusted network. Mostly, studies shed light on the collaboration within the enterprise boundary by isolating their employees from the rest of the world. Some works have identified the significance of collaboration not only in the enterprise environment but also across the enterprise. In [16], authors analyze and compare the existing vocabularies as a promising source for expert finding framework. To make the finding simple and structure, they highlight several factors such as common machine readable formats, reusable vocabularies and support of enabling technologies for practical use cases. In [17], authors raise the advantage of using linked data as an evidence source of expertise by analyzing the traditional information retrieval approaches. They also described some disadvantages of the linked data on the basis of the results of their hypothesis. Marian Lopez proposed a PeopleCloud platform [18] that enables experts to collaborate from inside and outside organization. The platform helps organizations in completing their tasks more efficiently and also leverage the expert networks for future activities. They illustrated the platform capabilities by discussing knowledge acquisition in IT inventory Management and IT support domain. Their comparison shows that the knowledge acquisition either explicitly or implicitly has significance to enterprises working environment. In [19], authors propose a propagation-based approach in order to find an expert in social network. They consider people local information as well as their relationship between people for their experiment. Their results show that the relationship is a useful factor for precision in expert finding. Capuano. N presents the enterprise framework by using semantic web technologies, assisting enterprises for collaboration [20]. Their framework comprises several layers and, each layer performs their own task. For instance, data representation handles modeling of data and data storage layer collects all the data from the data representation layer. In [21], authors propose the secure collaboration platform for enterprises by pointing out the security requirements for the cloud environments. They employ web service policy framework for their platform as a service (PaaS) infrastructure in order to mitigate the security threats. The aforementioned studies are insufficient in dealing privacy challenges in collaborative enterprises environment. In this paper, our objective is to propose a framework that will address the privacy issue in collaborative enterprise environment. Furthermore, the framework is not only capable of managing semantic-enhanced user profiles but also provide open innovation mechanism, which assists enterprises in collaborating new idea and finding relevant partners having right expertise.

III. Collaborative Enterprise Computing Approach

Enterprises is growing and expanding their businesses globally where different people from different geographical locations connect, communicate and collaborate for achieving their business goals. In this scenario, enterprises require reduction in the cost of IT infrastructures without compromising their business values. Cloud computing assists enterprises on-demand resources provisioning where enterprises can exploit different cloud computing models such as Platform as a Services (PaaS), Infrastructure as a Services (IaaS) and Software as a Service (SaaS) according to their conditions for reducing the IT costs and increasing the productivity. Microsoft, IBM and Google are notorious cloud computing providers not only providing data and network infrastructure to the enterprises but also providing software and applications for ease of business work. For instance, word processing, document management, content management and spreadsheets are delivered to enterprises on-demand without buying and installing into their enterprise environment. Indicators show that enterprises are considering the adoption of cloud computing in their environment and its market is growing with estimate of approximately $60 billion by 2012 [22].

Cloud providers have already appraised value of collaboration by incorporating social computing into their services and application, which opens a new horizon of innovation. Such collaborative environment facilitate enterprises in two ways: 1) It allows people to share their knowledge and information between partners and co-workers 2) It captures feedback from customers about products and services. In this way, enterprises can make their business processes efficient by involving skillful and competence people in the right place at right time and improving the quality of products and services rapidly by getting the response from customers. Hence, the overall impact will be increased efficiency and agility in the enterprises working environment that could lead to the introduction of new services and products to the market.

With the globalization of businesses, enterprises are producing large volume of disparate data with a different format, which are located on different geographical locations. In the

larger interest of enterprises, such data require to be exposed to different trusted partners and co-workers. The exposure can be done on the basis of the relationship between enterprises and/or people. The semantic web technologies can be used as the glue that helps in providing meaning and linking to enterprise data, services and user profiles. With such semantic enhance descriptions it is possible to employ vertical search on a predefined topic to get relevant and precise search results. The in-built reasoning capabilities of semantic web enables the system to deduce new facts from the existing facts. Today, many enterprises are adopting semantic web technologies into their software development life cycle to bring intelligence and smartness in the decision-making process. The semantic web technologies are being implied in many areas such as enterprise information integration, content management, life sciences and e-government. According to the gartner, the user of semantic web technologies in corporate, called as corporate semantic web, will reduce costs and improve the quality of content management, information access, system interoperability, database integration and data quality [23].

In our vision, we amalgamate cloud computing, social computing and semantic web technologies to expand the collaborative environment across enterprises boundaries and we commonly referred to Collaborative Enterprise Computing (CEC) as depicted in Figure 1. This fusion benefits enterprises

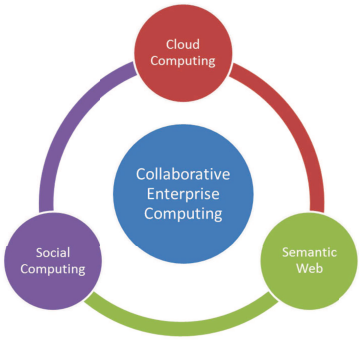


Figure 1: Collaborative Enterprise Computing

in many ways but low IT costs, correlate data of different enterprises and providing communication mediums (Blogs, Wikis, Social Network sites) are the most significant. The main rationale behind the CEC is to bring innovation through collaboration. Moreover, CEC ease the process of innovation by finding the trustworthy partners across enterprise boundaries who can be involved in the innovation process. These trustworthy partners are discovered/find according to their competences and experiences on the basis of criteria which can be given by enterprise.

IV. Case Study - Innovation Stock Exchange

The objective of developing the innovation framework is to create a trusted network for collaboration in an open-controlled environment. Collaboration allows enterprises, governments, entrepreneurs, academia, and other business entities to come together from a closed environment to an open environment and create new ideas as depicted in Figure 2.

Figure 2: Innovation through collaboration

This innovation ecosystem fosters the innovation process and introduces new products and services to the market. Such members of the innovation ecosystem could have the opportunity to register their ideas and openly collaborate with people of their trusted network. This could also lead towards the innovation stock exchange as depicted in Figure 3 where investors could invest on highly ranked ideas for increasing the business opportunities. By leveraging the collaboration into

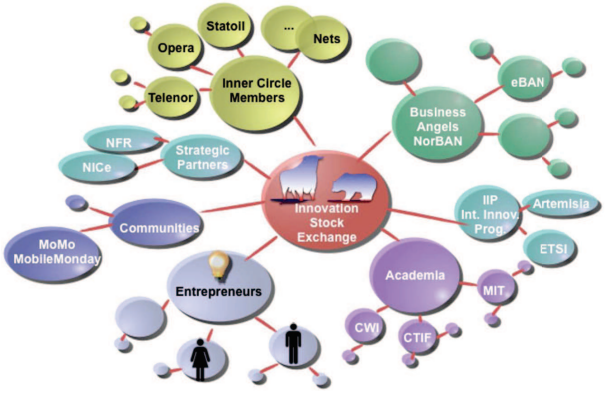


Figure 3: Innovation stock exchange a case study under consideration

the innovation ecosystem, enterprises will be able to develop innovative products and reduce their operational costs. For instance, Norwegian oil Industry reduced operational costs from 30-50% and enhanced productivity from 5-15% by integrating several operations together into their system [24]. Moreover, Procter&Gambler and Orange both has taken the initiative of collaboration by inviting people to present their ideas on a specific problem, and the most prominent idea was selected for the transformation into product [25]. Thus, significant revenue and customer satisfaction were acquired by both companies with innovation through collaboration. Current approaches are based on selected people from different organizations working together for a common goal. Our privacy-enhanced innovation framework will allow members of the innovation ecosystem to register an idea, assign scores from experts and find out trustworthy partners who can help in fostering the innovation

process so that the results can be achieved in the minimum time.

Assume Bob has an innovative idea, which he registers it in the system so that he could get right partners who could collaborate to transform the idea into realization. Idea will be reviewed for acceptance by the experts that belong to different enterprises. The idea will be published categorically according to the nature of its topic. The system will find experts in that topic by employing idea owner polices and criteria. The system will send notification to experts via email or sms and consider them to assign scores. The system facilitates Bob in finding trustworthy partners with whom he could collaborate for the realization of the idea regardless enterprise premises. Furthermore, the high-score idea will be published in the innovation stock exchange where members could open a vital investment opportunity. Thus, the system not only helping Bob in finding the right partner for his idea but also providing him implicit technical review, scientific value and the importance of his idea.

V. Enterprise Collaboration Enablement

This section outlines how an automated and trusted collaboration can be enabled in an enterprise.

A. User Profiling

User profile plays an important role for enabling automated collaboration beyond enterprise perimeter. User profiling generally involves profile setup, manipulation, and synchronization. In profile setup a basic user profile is created with explicit user feedback. The profile setup procedure can get user social network sites membership information from user basic profile, allowing profile setup mechanism to retrieve more information about user's preferences, groups and friends. In turns, this leads to implicit user feedback, where user information is collected without any intervention of user. The profile manipulation consists of create, read, update, and delete functions. The profile synchronization keeps update of all distributed profiles. ETSI [26] , 3GPP GUP [11] and MAGNET Project [27] among others are first initiatives towards standardization of user profile structure. However, these aforementioned research initiatives do not aim for collaboration. They mostly focus on personalized services. Whereas, in this outlook we specifically focus on user profile in the context of enterprise collaboration. We extend the user profile ontology proposed in [35], which classify the profile into different categories. Each profile contains relevant information according to its category and comprises authorization policy to restrict its access to third parties. For instance, corporate profile contains person's professional skills and expertise in a specific topic. This profile can only be accessed by third parties (i.e., colleagues, friend from trusted-virtual company etc.) to whom the permission has been granted. In this manner, a person can explicitly choose what to share and with whom to share his profile. Currently, we have defined one core concept Profile, which contains subclasses: (i) personal Profile, (ii) social profile, (iii) corporate profile, (iv) public profile and (v) private profile of it. A simplified snapshot of the user profile ontology is depicted in Figure 4.

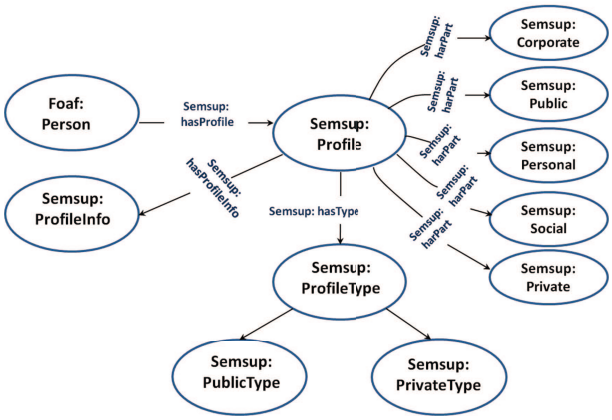


Figure 4: User Profile ontology

B. Trust Modeling

While Trust is relative term, which is defined differently in literature according to the nature of the work. In [28], author defines "Trust in a passionate entity is the belief that it will behave without malicious intent". In [29], authors consider context as an important factor for establishing trust by defining, "Trust is the firm belief in the competence of an entity to act securely, dependably and reliably within a specific context". In this paper, enterprise establish trust between their partners and co-workers to collaborate with each other for improving the quality of work and minimizing the risk factors. Thus, we define trust in such a way where the trusting agent has belief on trusted agent capabilities (see Figure 5) on the basis of relationship with the trusted agent for collaboration in order to realize a specific business opportunity.

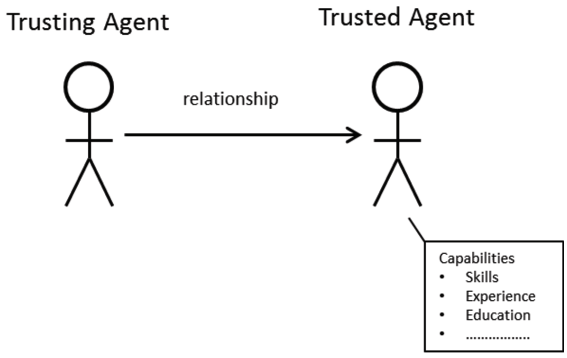


Figure 5: Definition of trust

We consider four factors context, time period, relationship and trust value that influence enterprises to obtain trusted partners for collaboration. The context is the situation or scenario for enterprises such as writing a research proposal, sharing new idea, discussing recent activity. Time period is a time at which one person interact with the other person and afterwards assign a trust value to it. For instance, one can

establish trust for the context ”writing a research proposal” in the time period 2009-2010. Relationship plays a pivotal role in the trust establishment, which associates the trusting agent with the trusted agent. The strength of the relationship is determined with the trust value which is assigned by trusting agent for a given context and time. Employees of an enterprise establishes a relationship with others by doing direct interaction and thus assign trust values to them. For instance, Bob meets Alice in a conference and becomes friend, Bob has a colleague and Bob meets Charlie on random meetings. Employees also receive recommendations about others from trusted friends, and trusted partners, which increase enterprises contacts not only within enterprise boundaries but also outside enterprise boundaries. For instance, Bob has a direct relationship with Alice, and Alice has a direct relationship with Charlie. Neither Bob nor Charlie has a direct relationship. Alice recommends Bob about Charlie and since Bob believes on Alice recommendation he can treat Charlie as a trusted partner. Trust relationship is depicted in Figure 6.

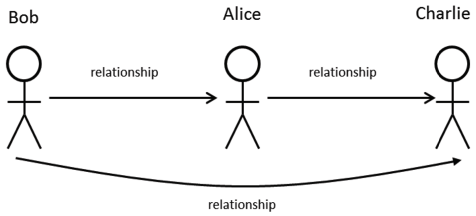


Figure 6: Trust relationship

To contrive social graph of trusted partners, Friend-of-a-Friend (FOAF) vocabulary helps for establishing friendship relation with foaf:knows property but it does not specify what is the value of friendship between two friends? FOAFRealm Onotology Specification [30] leverages the foaf:knows property by assigning the friendship value to the relationship. However, the ontology lacks in associating the value with given context and time period that are pivotal factors for enterprises. Thus, we propose to reuse FOAFRealm Ontology in conjunction of our Trust Ontology that allows employees to define their list of partners and assign trust values to them in a given context and a time period. The values can be given in the range of 0% (very distant) 100% (very close). The

Relationship	Trusting Agent	Trusted Agent	Time Period	Context	Trust Value
R1	Bob	Alice	2009-2010	Writing Research Proposal	90%
R2	Alice	Charlie	2003-2011	Working on project	95%

above table presents the two relationships R1 and R2 where Bob has 90% trust on Alice in the context of ”Writing research proposal” for the time period 2009-2010. This shows that Bob only trusts Alice in writing research proposal context and he does not trust her in other contexts. It can also be possible for Bob to assign trust values to Alice in different context for different time.

The trust Ontology is designed in [36] by considering the key elements (i.e., Trust direction, Trust Value, and Trust Type) to define the concept of Trust. Person can assign numerical trust values to other person with respect to their relationship. Furthermore, person can also assign multiple trust values to same person on multiple contexts. All this information is stored in Trust Ontology, which later can be used as a security attributes for assigning authorization policies to the user profiles. Moreover, we defined the concept of the TrustedParties as a union of ServiceProviders and Friends class and then subsume it to TrustedParties. A simplified snapshot of Trust ontology is depicted in Figure 7.

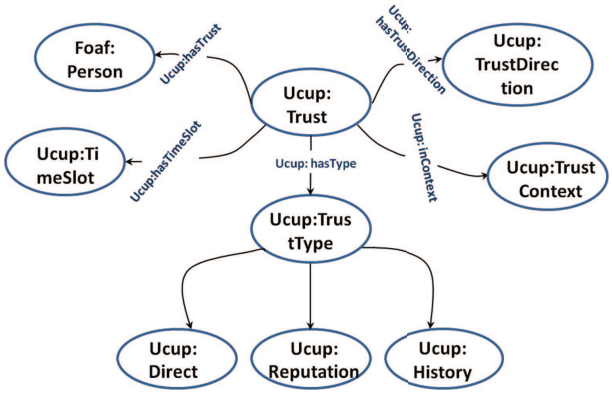


Figure 7: Overview of the trust ontology

C. Collaboration Criterion

As we discussed before that user-centric is one of the most demanding and prevailing feature of any collaboration platform, where innovation process is under perceived control specific innovation activity initiator. This can be achieved by defining collaboration criterion, where initiators can specify their conflict of interest, policy for establishing trust and some other requirements for automated collaboration. This paper proposes Semantic Web Rule Language (SWRL) [31], which is a combination of RuleML and OWL-DL [32]. In SWRL, rules are expressed in terms of OWL concepts i.e., classes, properties, individuals and literals. Rules are written in the form of Horn clauses antecedent (body) and consequent (head) where implication combines both the antecedent and consequent together. SWRL expressivity can be expanded with built-ins that provide traditional operations for comparison, mathematical transformation and URI construction. SRWL also enhances the expressivity by taking OWL expression (i.e., restrictions) in the antecedent or consequent of a rule but at the cost of undecidability. However, the undecidability issue can be resolved with DL-Safe rules [33]. The DL-Safe rule binds only known instances in ontology to rule variables. This restriction is sufficient to make SWRL rules decidable.

D. Business Idea Ontology

The Business Idea Ontology provides a mechanism to describe an idea which can be created by person,

reviewed by executive members of an organization and made it available for others in order to assign a score. The ontology is combined with existing ontologies, such as SKOS and FOAF, to achieve the modularity approach. We choose a hierarchical model that links our main classes id:Idea, org:ExecutiveMembers, skos:Concept, id:Score and foaf:Person to the super class owl:Thing. An Idea class (id:idea) contains the ideas by including abstracts, dates, keywords and title to it. ExecutiveMembers class contains the list of members who are responsible for providing review and assign scores. These scores reside in the subclasses of Score class (id: Score), which describes the assigned values in three terms such as ”Excellent”, ”Good” and ”Fair”. Furthermore, score class is created as a value partition class that included the subclasses ”Excellent”, ”Good” and ”Fair” as shown in the class definition.

$$Score \equiv Excellent \cup Good \cup Fair$$

We make these subclasses disjoint so that an individual cannot be a member of more than one class. In this manner, an idea can be classified on the basis of assigned score. We also defined properties (object and data) that allow us to describe the relationship between individual and literal values to these classes.

- id:hasTopic is an object property that links idea to the skos:Concept, describing the topic of an idea, e.g., Security, and Mobile Development.
- id:isCreatedBy is an object property that links the idea to a foaf:Person who is the creator of the idea.
- id:hasAssign is an object property point to the score class, containing score values that assigned by executive members.

The Figure 8 represents the complete overview of the Business Idea Ontology.

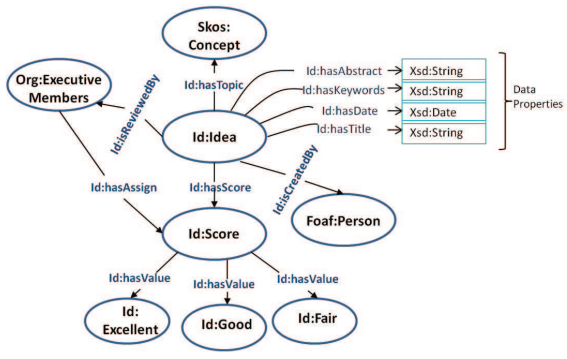


Figure 8: Overview of the business idea ontology

VI. Innovation Framework

The innovation framework is designed with the following components: Access Manager, Collaborate Ensembler and Collaboration Criterion Manger as depicted in the Figure 9. A framework first registers a person through Access Manager and allows him to create his user profile. After registration, the person interacts with Idea manager for the creation of a new Idea.

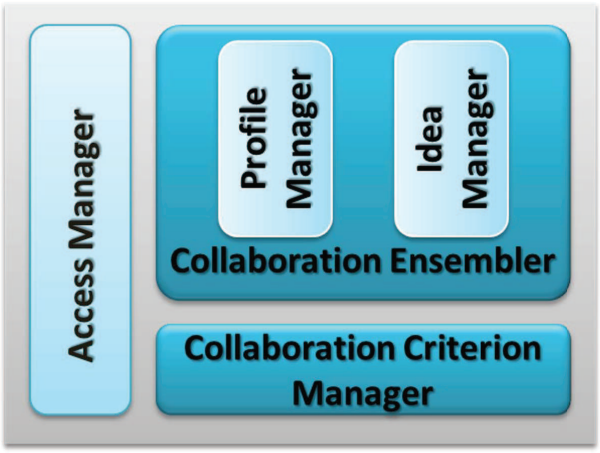


Figure 9: Innovation framework functional architecture

Idea manager notifies the members of an organization about the new idea so that they can review it and score it. After scores, the idea manager makes it available to different members or trusted-virtual communities according to their access rights that are accorded by the idea creator. Later, Collaboration ensembler reads the relevant information from the user profile and the idea along with the criteria from collaboration Criterion Manager by discovering the relevant partners. Apart from that, user profile manager also links the distributed user profiles by enabling linked data repository. This empowers a person to separate his corporate profile from his social or public profile and accord access according to their relationship. In this manner, person can expose his data in a controlled fashion where everything is under perceived control.

A. Profile Creation Phase

During the user profile creation phase, access manager receives a profile creation request from a user. First, access manager validates the user identity and after successfully validating the user, a profile creation page will be displayed where the user supply his information, needed by the profile manager. After completing the profile, access manager sends the CreateProfileRequest to Profile Manager, which stores the user profile information in the profile Knowledge Base (KB). Once the access manager receives the acknowledgment from the profile manager, it sends the ProfileCreatedResponse to the user as depicted in Figure 10. The user can also be asked to present the URI of his distributed profiles so that the user profiles can be integrated from multiple sources. These linked user profiles are stored in the linked data repository and later, co-workers or other third parties can access accordingly.

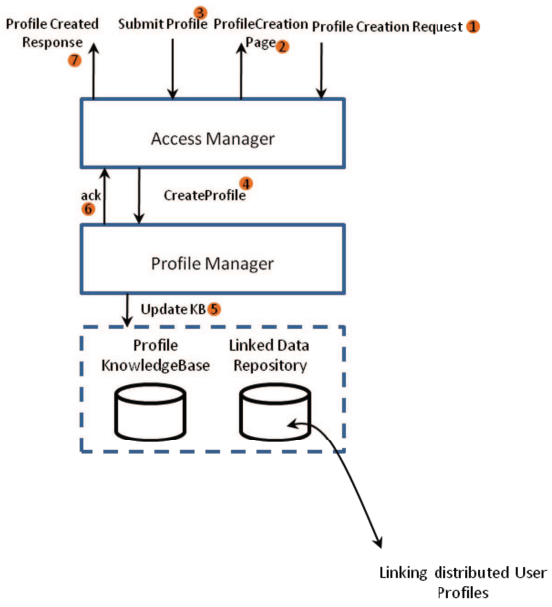


Figure 10: Profile creation phase

B. Idea Registration Phase

The idea manager is responsible for managing the idea requested by the Access Manager. Idea manager is also responsible by providing the mechanism of assigning scores. Initially, the person requests Access Manager for the registration of a new idea. After successfully validating the identity of the person, access manager precedes him to the IdeaRegistrationPage where idea can be written by providing its Title, Abstract and Date. Idea manager stores the idea in to the Idea Knowledge Base (KB) upon receiving RegisterIdea request from the Access Manager by getting SubmitIdea request from the person as depicted in Figure 11. Once the idea is registered

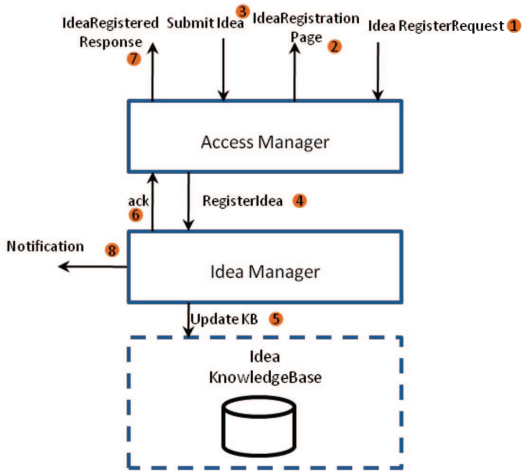


Figure 11: Idea registration phase

and stored in the knowledge base, Idea Manager sends the notification through SMS or email to executive members of the organization so that they can make the innovation process effective by involving themselves as soon as possible.

C. Score Phase

After receiving the new idea registration notification, the executive members can review the idea. To initiate the review process the executive members provides their credentials and idea name to access manger, which in response return the newly register idea review page after validating the credentials. The executive member can submit their score after reviewing the idea. Once all the executive members submit their score the IdeaManager calculate the overall score and set the status of the Idea based on the score. If the idea achieved status of open for collaboration then the CEE exploits description logic [34] based reasoning capabilities over user profile KB and the approved idea by incorporating collaboration criteria associated with the idea. The end result of this reasoning process is a trusted-virtual company, containing a list of relevant partners that are suitable for the approved idea. The score phase in depicted in Figure 12.

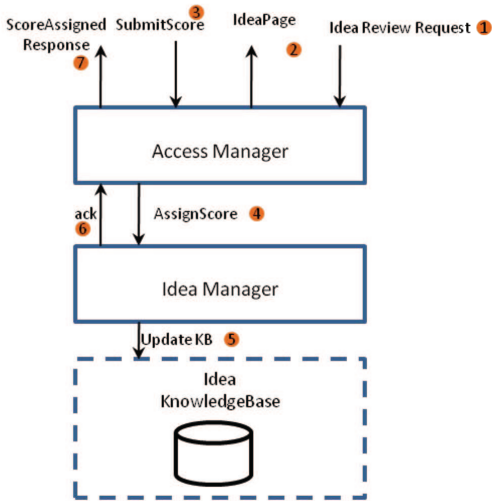


Figure 12: Score phase

VII. Implementation Road Map

Having described the innovation framework architecture, this section outlines the road map for implementation of cloud based services using the state-of-the-art technology enablers.

We propose to implement innovation framework in the form of APIs as they are becoming mainstream. The essence of this approach is better integration with existing apps, enablement of custom apps development and augmentation of existing apps with new functionalities. Additionally, we favor an open API strategy instead of an internal-first API strategy where APIs are developed internally first and then shared with close partners and in the last phase made them open to the world.

This outlook aims to bring all stakeholder from all areas of collaborative ecosystem, including industry and academia, to enhance and ease off innovation process. Such inter-organization collaboration demands a common/shared place to publish and share novel and innovative ideas without

delving into technology infrastructure. We anticipate that cloud computing platform is the most appropriate for such inter-organization collaboration because it allows to focus more on delivering services rather than on managing technology infrastructure. We use cloud platform as a service endpoint provider and data storage. In this regard, we select the windows azure platform [37], which comprises of: (1) Windows Azure - an operating system as a service that facilitate on-demand compute, storage and mange web application on the internet, (2) SQL Azure - a relational data storage service in the cloud that foster reuse of familiar relational models, tools and utilities, (3) Windows Azure AppFabric - a cloud-based infrastructure services for applications running in the cloud or on enterprise premises.

The prototype implementation of innovative stock exchange case study can be realized in the form of windows azure services. For this purpose, we propose to develop UserProfileService, IdeaRegistrationService, IdeaRatingService, and IncubationService by using the innovation framework APIs. Each service is backed by a DB storage such as ProfileDB, IdeaDB, and ScoreDB. Despite the fact that windows azure platform provides a wide range of storage options but it still lacks the support of semantic enhance storage (i.e., triplet storage). This limitation can be fixed by having a mapping mechanism for proposed ontologies that is capable of incorporating ontology level changes into relational storage. Such mechanism can maintain semantic related stuff into separate tables for each service DB storage, which works as an overlay for the each service DB storage. These windows azure services can be integrated with other apps regardless of the technology since windows azure supports different standards, protocols and languages including REST, SOAP, JAVA, PHP and Ruby. However, we will focus only on SharePoint Server 2010 [38] (i.e., Microsoft based enterprise collaboration and social computing platform). The integration of windows azure services with SharePoint 2010 requires the development of Silverlight enabled Web Parts. In this case, each Web Part is associated with some Windows Azure service and SharePoint acts as service consumer. The overall integration strategy is depicted in Figure 13.

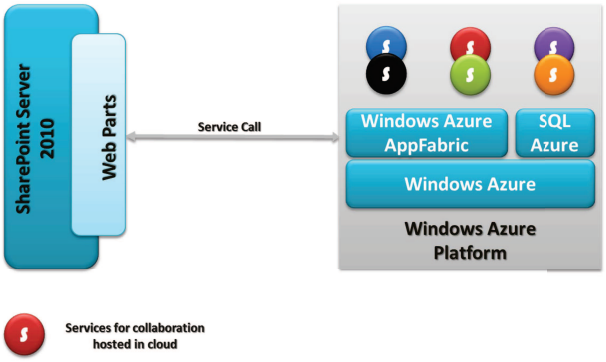


Figure 13: Integrating collaboration services with SharePoint server 2010 using SharePoint Web Parts

VIII. Conclusion and Future Work

In this paper, we proposed the collaborative enterprise computing (CEC) approach, helping in the creation of the network of enterprises for enabling active, automated and trusted inter-enterprise collaboration. This is achieved by considering the challenges of user profiling, balancing of open and close collaboration and trust establishment. The CEC approach is quite significant for discovering trusted relevant partners who could involve in the innovation life cycle process.

The proposed framework comprises three core components such as Access Manager (AM), Collaboration Ensembler (CE) and Collaboration Criterion Manager (CCM). As the framework is designed by considering the standard semantic web tools it inherits some built-in features such as interoperability, integrating of data from multiple sources, and reasoning for deriving the entailment facts from the knowledge base. We also designed semantically enriched user profile ontology, trust ontology and business idea ontology by considering the modular approach. Moreover, the paper provides the road map for the implementation of the innovation framework in the form of APIs. The API oriented approach is suited for better integration with other apps. We proposed to develop cloud based services such as UserProfileService, IdeaRegistrationService, and IdeaRatingService using proposed APIs. Our exploration shows that capturing enterprise employee expertise, and ideas in a structured and machine understandable way are highly eminent for an automated inter and intra- enterprise collaboration.

Our ongoing and future work includes evaluating the framework by describing the sophisticated criteria for discovering relevant partners. We are also considering enhancing the framework by providing a Trust Management component, which can ease the trust assigning and evaluation process. Moreover, we will evaluate the framework in a real environment.

Acknowledgment

This work is supported by the Norwegian Research Council.

References

[1] P. Banerjee, R. Friedrich. L. Morell, ”Open innovation at HP Labs”, IEEE Computer Society, Nov. 2010, pp. 88-90, doi:10.1109/MC.2010.322.

[2] J. Bughin, M. Chui, and A. MillerHow, ”Companies are benefiting from Web 2.0: McKinsey Global Survey Results”, http://www.mckinseyquarterly.com/Business_Technology/BT_Strategy/How_companies_are_benefiting_from_Web_20_McKinsey_Global_Survey_Results_2432 [accessed on 31 January 2011]

[3] Oracle, ”The Social Enterprise: Using Social Enterprise Applications to Enable the Next Wave of Knowledge Worker Productivity”, http://whitepapers.techrepublic.com.com/abstract.aspx?docid=391431 [accessed on 31 January 2011]

[4] A. Fu, C. Finn, D. W. Rasmus, R. Salkowitz, ”Social Computing in the Enterprise Microsoft vision for Business Leaders”, Microsoft White Paper, 2009

[5] K. Efta, ”Enterprise Social Computing”, http://www.allyis.com/thinking/Pages/Enterprise-Social-Computing.aspx [accessed on 31 January 2011]

[6] A. P. McAfee, ”Enterprise 2.0: The Dawn of Emergent Collaboration”, http://sloanreview.mit.edu/the-magazine/articles/2006/spring/47306/enterprise-the-dawn-of-emergent-collaboration/3/ [accessed on 31 January 2011]

[7] D. Hinchcliffe, "Top Ten issues in adopting Enterprise Social Computing", http://www.zdnet.com/blog/hinchcliffe/ten-top-issues-in-adopting-enterprise-social-computing/581 [accessed on 31 January 2011]

[8] G. Antoniou, F. Van Harmelan, "A Semantic Web Primer", MIT Press 2008.

[9] FOAF Project, "http://www.foaf-project.org/", [accessed on 31 January 2011]

[10] R. Gosh, M. Dekhil, "Discovering User Profiles", Proceedings of the 18th international conference on World wide web, 2009.

[11] 3GPP TS 29.240, "3GPP Generic User Profile, Stage 3, Release 6", http://www.3gpp.org/ftp/Specs/html-info/29240.htm [accessed on 31 January 2011]

[12] D. Heckmann, E. Schwarzkopf, J. Mori, D. Dengler,A. Kroner, "The user model and context ontology GUMO revisited for future Web 2.0 Extensions", Contexts and Ontologies: Representation and Reasoning, pp.37-46

[13] D. Heckmann, A. Krueger, "A user modeling markup language (UserML) for ubiquitous computing", User Modeling 2003 In User Modeling 2003 (2003), pp. 148-148

[14] J. Stan, E. Egyed-Zsigmond, A. Joly, P. Maret, "A user profile ontology for situation-aware social networking" , 3rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI2008)

[15] IST FP6 Spice project,"http://www.ist-spice.org/" [accessed on 31 January 2011]

[16] B. Aleman-meza , U. Bojars , H. Boley , J.G. Breslin, M. Mochol , L. Jb Nixon , A. Polleres , A. V. Zhdanova, "Combining RDF Vocabularies for Expert Finding", in proceedings of the 4th European Semantic Web: Research and Applications , 2007

[17] M. Stankovic, C. Wagner, J. Jovanovic, P. Laublet, "Looking for Experts? What can Linked data do for you", in proceedings of the Workshop on Linked Data on the Web, April 27, 2010

[18] M.Lopez, M. Vukovic, J. Lardeo, "People Cloud Service for Enterprise Crowdsourcing", in proceeding of IEEE Service Computing, 2010

[19] J. Zhang, J. Tang, J. Li , "Expert Finding in a Social Network", in proc. of Advances in Databases: Concepts, Systems and Applications In Advances in Databases: Concepts, Systems and Applications, Vol. 4443 (2010), pp. 1066-1069-1069.

[20] N. Capuano, M. Gaeta, F. Orciuoli, P. Ritrovato, "Semantic Web Fostering Enterprise 2.0" in Proceeding of Intelligent and Software Intensive Systems, International Conference, Los Alamitos, CA, USA, 2010.

[21] S. Bertram, M. Boniface, M. Surridge, N. Briscombe, M. Hall-May, "On-Demand Dynamic Security for Risk-based Secure Collaboration in Clouds", in Proceeding of IEEE 3rd international conference on Cloud Computing, Miami, August 2010.

[22] Cisco White Paper, "Transforming Enterprise IT Services with a Secure, Compliant Private Cloud Environment", http://www.cisco.com/en/US/services/ps2961/ps10364/ps10370/ps11104/services_cloud_enablement_white_paper_enterprise.pdf [accessed on 15 January 2011]

[23] Gartner Press Release, "Gartner's 2006 Emerging Technologies Hype Cycle Highlights Key Technology Themes", http://www.gartner.com/it/page.jsp?id=495475 [accessed on 5 January 2011]

[24] R. A. Fjellheim, R. B. Bratvold, M. C. Herbert, "CODIO - Collaborative Decision making in Integrated Operations", in proc. of Intelligent Energy Conference and Exhibition, Society of Petroleum Engineers, 2008.

[25] NESTA, "Open innovation from marginal to mainstream", http://www.nesta.org.uk/library/documents/Open-Innovation-v10.pdf [accessed on 5 January 2011]

[26] ETSI User Profile Management, http://portal.etsi.org/stfs/STF_HomePages/STF265/STF265.asp [accessed on 15 January 2010]

[27] MAGNET Project, http://www.telecom.ece.ntua.gr/magnet/index.html [accessed on 15 January 2011]

[28] A. Jsang, "The right type of trust for distributed systems," in Proc. of the 1996 workshop on New security paradigms (NSPW '96). ACM, pp. 119-131, doi:10.1145/304851.304877, http://doi.acm.org/10.1145/304851.304877.

[29] T. Grandison, M. Sloman, "A Survey of Trust in Internet Applications," in proc. of the Communication Surveys and Tutorials, IEEE, Volume 3 Issue 4, pp. 2-16, 2000, doi:10.1109/COMST.2000.5340804.

[30] FoaFRealm Ontology Specification, http://www.foafrealm.org/xfoaf/0.1/index.html}accessed on 15 January 2011]

[31] Semantic Web Rule Language, http://www.w3.org/Submission/SWRL/ [accessed on 15 January 2011]

[32] Ontology Web Language, http://www.w3.org/TR/owl-guide/ [accessed on 15 January 2011]

[33] B. Motika, U. Sattlerb, R. Studera, Query Answering for OWL-DL with Rules, in Web Semantics: Science, Services and Agents on the World Wide Web journal. Volume 3, Issue 1, July 2005, Pages 41-60.

[34] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider "The Description Logic Handbook: Theory, Implementation and Application", Cambridge University Press, 2002.

[35] Zahid Iqbal, Josef Noll, Sarfraz Alam, Mohammad M. R. Chowdhury, "SemSUP: Design and Implementation of Semantic Enhance Social-Aware User Profile", in proc. of IEEE DEST, 12-15 April 2010, Dubai.

[36] Zahid Iqbal, Josef Noll, Sarfraz Alam, Mohammad M. R. Chowdhury, "Toward User-centric Privacy-aware User Profile Ontology for Future Services", in proceedings of IEEE third International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ 2010), 13-19 June 2010, Athens, Greece, pp. 249 - 254.

[37] Windows Azure - Microsoft's Cloud Services Platform, http://www.microsoft.com/windowsazure/ [accessed on 15 January 2011]

[38] Microsoft SharePoint Server 2010, http://sharepoint.microsoft.com/en-us/Pages/default.aspx [accessed on 15 January 2011]

Fault Tolerance Framework using Model-Based Diagnosis: Towards Dependable Business Processes

Angel Jesus Varela-Vaca, Rafael M. Gasca, Diana Borrego, Sergio Pozo
Computer Languages and Systems Department,
Quivir Research Group
ETS. Ingeniería Informática, Avd. Reina Mercedes S/N,
University of Seville, Seville, Spain
{*ajvarela, gasca, dianabn, sergiopozo*}*@us.es*

***Abstract*—Several reports indicate that one of the most important business priorities is the improvement of business and IT management. Management and automation of business processes have become essential tasks within IT organizations. Nowadays, business processes of a organization use external services which are not under our its jurisdiction, and any fault within these processes remain uncontrolled, thereby introducing unexpected faults in execution. Organizations must ensure that their business processes are as dependable as possible before they are automated. Fault tolerance techniques provide certain mechanisms to decrease the risk of possible faults in systems. In this paper, a framework for developing business processes with fault tolerance capabilities is provided. Our framework presents various solutions within the scope of fault tolerance, whereby a practical example has been developed and the results obtained have been compared and discussed. The implemented framework presents innovative mechanisms, based on model-based diagnosis and constraint programming which automate the isolation and identification of faulty components, but it also includes business rules to check the correctness of various parameters obtained in the business process.**

Keywords-Business process, Business Process Management, Fault-tolerance, Dependability.

I. INTRODUCTION

The automation of business processes is emerging into the enterprise arena as a mechanism for improvement. Companies may decide to deploy Business Process Management System (BPMS) to automate their business processes, but it system cannot guarantee perfect executions error-free or fault-free execution. On the other hand, nowadays there exist a trending in the integration of services of different companies in the business processes. The integration of external services set up new point of vulnerability in the business process execution. Companies have to ensure that their business processes are as dependable as possible using mechanisms such as introduced in [1][2]. Gartner's CIO report [3] indicates that the most important business priorities include: improvement of business processes, cost reduction, enterprise workforce effectiveness, security and IT management. Therefore, dependability is a significant requirement for many types of companies, since any failure in their business processes may lead to terrible consequences: economic lost, lives lost, systems destroyed, security breaches, and so on.

In recent years, a new paradigm has emerged in the scope of business IT: Business Process Management (BPM). BPM is defined as a set of concepts, methods and techniques to support the modelling, design, administration, configuration, enactment and analysis of business processes [4]. BPM has become an essential tool for organizations, since it is defined as a methodology for the improvement of the efficiency through systematic management of business processes that should be modelled, automated, integrated, monitored and optimized in a continuous way. One of the most important goals of BPM is the better understanding of the operations that a company performs and the relationships between these operations. BPM also aims at narrowing the gap between business processes that a company performs and the implementation of these processes in the BPMS.

The BPM paradigm follows a life cycle that consists of several stages [5], shown in Figure 1. During each stage, various kinds of faults can be introduced:

- In the design stage, business process models can present some design faults (such as deadlocks, live-locks and starvations). Some systematic approaches provide design guidelines that allow their designed processes to be corrected and improved. Design problems are not taken into account in this paper since it is an issue that has already been subject to wide discussion [6][7][8].
- In the run-time stage, faults could be located in the business processes when unexpected outputs, unexpected messages, unexpected events, or unexpected performances are obtained. Executable business processes tend to use external services that are not under their jurisdiction. Thus, it is impossible to ensure that the functionality of a certain external service changes during the business process life cycle. As a consequence, unexpected changes in services could entail changes in business process behaviour.

Therefore, companies must pay attention on the inclusion

of measures that promote the reduction of the risk of possible faults and increase the dependability of business processes from design stages. The majority of proposals have used fault tolerance ideas in other areas such as: firewalls, grid computing, composition of applications and service-oriented architectures, [9][10][11][12][13]. In these studies, various fault tolerance approaches have been applied: check-point view [9], replication and recovery techniques [10][11], and other sophisticated techniques such as dynamic binding [12], and self-reconfiguration of systems [13]. Our approach proposes to improve dependability properties in the execution of business processes based on techniques for automatic identification of faults by means of model-based diagnosis which helps to establish specific fault tolerance mechanisms. Fault tolerant mechanisms applied in this work are based on classic fault tolerance ideas such as replication, check-pointing and techniques of diversity focused on the context of service-oriented business processes.

This paper is structured as follows: Section II introduces some concepts of BPM and fault tolerance; Section III presents our framework for dependable business processes using fault-tolerant techniques; in Section IV, a practical example is explained and developed; Section V shows experimental results that are discussed; and, in the last section, conclusions are drawn and future work is proposed.

II. Business Process Management System and Fault Tolerance

In order to understand the BPM paradigm, it is necessary to show the typical business process life cycle [5], as shown in Figure 1. The life cycle consists of different stages:

1) **Design and Analysis:** business process models are defined and validated.
2) **Configuration:** once business process models are validated, they need to be implemented by a dedicated software system. In this configuration stage a software systems (BPMS) is chosen and configured where business processes will be deployed.
3) **Enactment:** once deployed, process enactment needs to guarantee correct execution in accordance with various constraints specified in the model.
4) **Diagnosis:** techniques are applied to identify and isolate faults in business processes. Nowadays, most diagnosis techniques applied are focused on identifying design faults.

In BPM, several levels of business processes can be identified depending on the point of view of the organization [4]. In this paper, only two levels of BPM are considered:

- Operational business processes are described with business process models where the activities and their relationships are specified, but implementation aspects are not taken into account. Operational processes are the basis for developing implemented business processes.

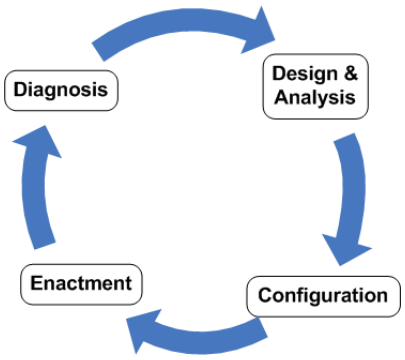


Figure 1. Business process life cycle.

- Implemented business processes contain information about the execution of the activities and the technical and organizational environment where they will be deployed and executed.

Implemented business processes and specifically service-based business process use external services which remain outside the jurisdiction of the organization. Although, business process models are validated and verified at the design stage, organizations cannot guarantee the correct execution of changes in services interms of functionality and faults simply through the lack of response or security attacks, and so on. The identification of where business processes are failing, and which components are involved in the faults may probe worthwhile for the numerous business stakeholders (designers, analysts, developers, etc).

Fault diagnosis is a method which permits us to determine why a correctly designed business process fail to work as expected. Diagnosis aims to identify and isolate the reason of any unexpected behaviour, or in other words, to identify which parts are failing in a business process. In this work, model-based diagnosis [14] is used in order to isolate the faulty services. Model-based diagnosis is recognized as a very powerful tool within the community of diagnosis due to its ability to solve the problem of isolating faulty components.

Our proposal is focused on achieving dependable business processes, but to the achievement of dependability must first be clarified [15]. The properties for achieving dependability fall into four major groups: fault tolerance, fault avoidance or prevention, fault removal, and fault forecasting.

By definition, fault tolerance [15] is a mechanism used to order to guarantee service by complying with the specification in spite of the presence of faults. Fault-tolerance techniques are a means of reducing the risk of faults. On the whole, fault-tolerance frameworks are focused on physical systems and not on software systems and most applied techniques are based on replication and recovery. Replication is employed in the recuperation of services by means of duplication of each of its functionalities in form of replicas

and in the case of a service replica fault another replica takes control. Typical solutions in replication are considered as:

- *Passive replication* [16]: the client only interacts with one replica (primary) which handles the client request and sends back responses. The primary replica also issues messages to the backup replicas (other secondary replicas) in order to update their state.
- *Active replication* [16]: all replicas play the same role. All replicas receive each request, handle the request, and send back the response to the client. Other solutions based on active replication [17] exist.

Initially, active replication provides a generally faster response time than passive replication. However, in active replication all replicas must to process the requests, and hence more system resources are employed than for passive replication. Moreover, the nested redundant replicas could cause a problem of invocations since the use of active replication requires replicas to be deterministic. No such determinism is required for passive replication, and therefore it is more flexible.

In the majority of cases, fault tolerance is used as only a hardware approach, but software is also a crucial factor in the effective good running of organizations. Increasing the dependability of software presents some unique challenges when compared to increasing the dependability of traditional hardware systems [18]. Hardware faults are mainly physical fault, which can be characterized and predicted over time. Software has only logical faults which are difficult to visualize, classify, detect, and correct. Changes in operational usage or incorrect modifications may introduce new faults. To protect against these faults, it is insufficient to simply add redundancy, as is typically done for hardware faults, since doing so would simply duplicate the problem.

III. FAULT TOLERANCE FRAMEWORK

In the previous section, the basic ideas of the business process life cycle are introduced and the main stages to manage business processes are shown. This section is focused on presenting the framework by defining all its components and clarifying how it works. The proposed framework is based on the main ideas of the BPM life cycle, whose structure is depicted in Figure 2. As can be observed, the framework is structured in four main layers: Modelling, Applications, Fault Tolerance, and Services. In the following subsections, the various parts of the framework are detailed and discussed. Although it can be noticed that the main characteristic is the utilization of a specific fault-tolerance layer. This layer contains specific mechanisms in order to mitigate the risk of possible process faults detected in the execution of business processes.

Before continuing any further, some assumptions about business processes should be stated:

1) A business process model has a single start event, a single end condition and every activity contributes towards finishing the process correctly.
2) The business process design is correct, and hence no design faults exist(deadlocks, live locks, starvations, and so on).
3) Operational business processes cannot suffer byzantine faults (arbitrary faults).
4) Operational business processes are stateless.

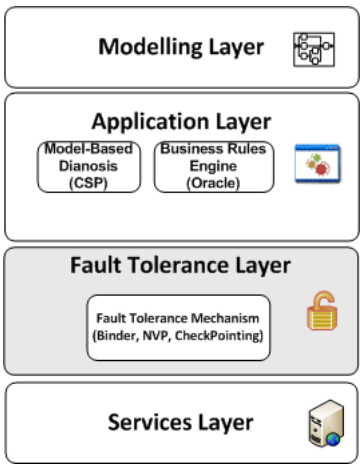


Figure 2. General view of the framework.

Therefore, the main problems left for discussion in this paper involve the diagnosis and mitigation of incorrect outputs/results and events in business processes. In the following subsections, each framework layer is described.

A. Modelling Layer

The design stage is focused on describing different models used in our framework. Our approach is based on three kinds of models:

- Business process models are graphical descriptions of business processes (BP). These models represent the graphical formalization of the various constraints to which a business process has to comply at any time they are employed. Within the arena of BPM, various modelling languages have emerged: Flowcharts, Petri Nets, Event-driven Process Chain (EPC), UML Activity Diagrams, Data Flow Diagrams (DFD), IDEF, and Business Process Management Notation (BPMN). In the design stage, the most widely used for business processes representation is currently the standard Business Process Management Notation (BPMN) by OMG [19]. This notation is highly useful for business analysts since BPMN contains many different elements such as activities, data objects, and gateways. However, BPMN diagrams save no information on where they will be executed and are platform-independent. Figure 3 shows an example of a BPMN diagram where elements are described. Our framework can support BPMN diagrams,

and a prototype is developed and shown in [6].

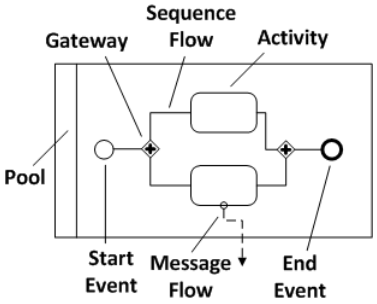


Figure 3. BPMN element description.

- Business rule models; a business rule is a statement that defines those aspects of business processes that are impossible to gather or express in the model [20]. In fact, by separating business logic from business design, if the business process logic changes then, only the business rules must be changed but not the business process. Business rules can be formalized as ”if-then” statements in a selected Business Rule Management Systems (BRMS), by using natural language or formal methods. BRMS is a software system employed to define, deploy, execute, monitor, and maintain the variety and complexity of decision logic that is used by operational systems within an organization. In our proposition, business rules act as an ”oracle”. Thus, they indicate the correctness of outputs of the business processes.
- Constraint models are necessary in the model-based diagnosis stage where Constraint Satisfaction Problems (CSP) are used. In our proposition, constraint models are employed to describe behaviour activities gathered within business processes. Constraint models can be constructed off-line or on-the-fly. Process constraint models can be built online since logs captured where information of inputs, outputs and service constraint model can be given. Both Diagnosis and CSPs will be described in Section III-C.

Business process validation analysis methods are focused on discovering whether the designed business processes can be automatically enacted as expected. Through this analysis, any possible constraint violations should be detected. In various studies, process models have been analyzed in order to prevent structural faults in the form of : Petri Nets [21]; a set of graph reduction rules to identify structural conflicts in process models [22]; and an improved version of the latter method [23]. However, in a earlier work [6] a business process validation has been studied, and an automatic mechanism to fault structural diagnosis has been integrated in a editor of BPMN diagrams.

B. Application Layer

In this layer, various technologies which are used to implement and deploy models are introduced. Business Process Execution Language (BPEL) is a de facto standard language for the implementation of service-based business processes. The main advantages in using BPEL as an implementation language are:

- BPEL supports all necessary elements in the implementation of sophisticated business processes, such as those in BPMN, and provides a sufficient number of mechanisms to implement fault tolerant mechanisms [24].
- BPEL processes are specific implementations of business processes for service-oriented environments, as suggested in [4].
- The majority of the distinguished commercial and non-commercial tools support service-oriented architectures and BPEL processes as shown in Table I.

Table I
COMPARATIVE OF BPM TOOLS.

	Process Modeling	Methodology	SOA	Support BPEL	Translation BP2BPEL
Intalio BPMS	√	BPMN	√	√	√
IBM WepSphere Process Integration	√	No Standard	√	√	√
Appian BPM Suite	√	BPMN	√	-	-
Tibco iProcess Suite	√	BPMN	x	x	x
PegaSystem SmartBPM Suite	√	-	-	-	-
Oracle BPM Suite	√	BPMN	√	√	¿
G360 Enterprise BPM Suite	√	BPMN	x	x	x
Lombardi BluePrint & Teamwork	√	BPMN	x	x	x
Savvion BusinessManager Platform	√	BPMN	√	√	x
Fujitsu Interstage BPM Suite	√	BPMN	√	√	x
IBM FileNet BP Manager	√	BPMN	√	x	x
Aura Portal BPMS	√	BPMN	√	x	x

Although BPMN is a standard notation for the design of process models, it could automatically be translated into BPEL [19][25][26], and most commercial BPMSs support BPMN and BPEL processes. Therefore, BPEL processes represent the best candidate and will be used in our proposal. A BPEL environment is necessary for the implementation and deployment operational business process. In this proposal, GlassFishESB have been integrated in our framework. NetBeans provides an environment in order to develop BPEL

processes graphically. Moreover, GlassFishESB provides support as an application server with several components of Enterprise Service Bus (ESB) [27] and a runtime for BPEL processes.

There is no accepted standard for business rule systems. For this reason, in our proposal business rules have been developed using WebSphere Ilog JRules and have been integrated within business processes as services.

Neither is there any standard CSP representation nor CSP Solver. For the development of constraint models, any CSP Solver could be used. In our case, constraint models have been implemented using ChocoSolver [28] which has been integrated into our framework. However, the model could be implemented as a standalone from other CSP Solver tools, and used as a service.

C. Diagnosis

In our approach, model-based diagnosis is applied. Diagnosis is used for the identification and isolation of behavioral faults in the components of business processes. Constraint Satisfaction Problem (CSP) techniques have been applied in the diagnosis since CSP is an extended technique which solves a wide variety of problems, including those in business processes [29]. In order to apply CSP, business processes have been transformed into Constraint Optimization Problems (COP) [30]. COPs are specific CSPs where an objective function to optimize remains to be optimized. COP is evaluated using a CSP solver (a solver is an engine of constraints that implements algorithms in order to solve constraint satisfaction problems). The diagnosis will only be invoked when a fault in the outputs of the business processes has been detected.

D. Fault Tolerance Layer

Framework has been built with a specific fault tolerance layer. This layer is developed with the intention of controlling possible faults and taking corresponding corrective actions in order to recover the execution business processes, and then to achieve dependable properties. In our case, some replication solutions have been adopted, but on the other hand, software fault tolerance techniques have also been considered. Likewise, replication and recovery techniques are focused on the redundancy of components without considering the business process state. Therefore, one fault tolerance mechanism is focused on the simulation of checkpoint and recovery [31], but in this case oriented towards business processes. Various fault-tolerant mechanisms are described below:

1) **Without fault tolerance**
   BPEL processes are defined as a composition of web services. Thus, web services (of either external or internal organization) are linked at design time. If a fault output or event occurs (supposing no design faults exist in the processes), this fault is located on the service side, for instance, a change of functionality, or a miss-match of parameters. Once the BPEL process has been deployed, it is impossible to replace the faulty service during run-time. Faults are only solved by stopping for every instance of a business process. Firstly, the faulty service has to be located (diagnosis), then eliminated, and finally replaced with another correct service. Therefore, it is necessary to introduce mechanisms to mitigate effects produced by faults without having to stop the execution of business processes and this can be carried out by means of fault tolerance techniques.
2) **Primary/Backup approaches**
   This solution applies the concept of redundancy, in the sense of replication of services. This approximation has a primary service as principal and one or more replicas as backups. In the case of a fault, it is possible to use the backup services. The adopted solution in the proposal is based on dynamic binding ideas [32][33]. Dynamic binding is a technique that allows services to be linked at run-time.
   In our approach, a binder component is introduced between the BPEL processes and the services acting as proxies, as shown in Figure 4. Binder is not developed as an external program or external monitor, but as another business process. The binder component decides at run-time what services to invoke: whether they be the original service or the replicas (backups). The solution is fault tolerant since in the case of detecting a faulty service, every faulty service invocation can be replaced with an invocation to a backup service.

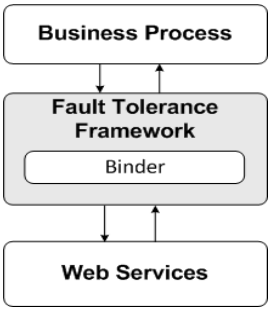


Figure 4. Communication process-service with binder.

This is a good solution despite presenting some deficiencies such as the introduction of a unique point of fault at binder component which in turn could introduce a very high overhead in the performance of a business process. In order to solve the first problem, the replication of the binder can be applied, thereby also, replicating the binder component (one primary and several backup replicas). In our case, passive replication is used in the proposed solution. In the case of a fault, in the first binder, a backup binder

takes the control of the execution, thereby enabling the execution to continue, see Figure 5. The binder backup can be an exact copy of the original but located in another external server or machine that of the primary copy. BPEL provides various mechanisms, such as fault and compensation handlers, to achieve the implementation of primary-backup binders.

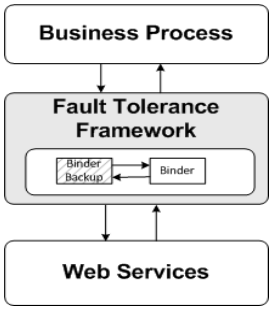


Figure 5. Communication process-service with binder replicated.

3) **Multi-Version (N-Version Programming) approach**
Software components cannot be degraded in the same way as physical systems and redundancy fails to provide a good solution, since if a fault is detected in a software component, it is due to an implementation fault with a very high probability, and this kind of fault cannot be solved by means of redundancy. Diversity is a very important factor in obtaining dependable software systems [18]. The main goal of diversity is to provide identical services (variants) but with a separate design and implementation in order to minimize causing identical faults. For instance, when a software variant presents a fault, this will be isolated as much as possible. There are different techniques for fault-tolerance software based on multi-version (diversity of software) [18][34]: N-Version Programming (NVP), Recovery Blocks, N-Self Checking Programming.
In the proposed framework, a solution based on NVP is adopted, which is a static technique where a activity is executed by various processes or programs and the result is only accepted by majority of votes. This mechanism for obtaining results is defined in NVP as an adjudicator or decision mechanism (DM). Various decision mechanisms are defined in [18].
The N-Version paradigm considers the utilization of diversity for implementations and designs in order to isolate faults in the components. Every service used in the implementation will be developed as an N-Version Component. The implementation selected for N-Version components follow the basic ideas of N-Version Programming. An N-Version component provides at most $2X + 1$ replicas, X ranges from 1 to N, and where N is an integer greater than 1. Components have been developed with an adjudicator (DM) in order to obtain the output results, see Figure 6. The logic within the adjudicator (DM) can be very basic or seriously complicated. However, NVP components can be improved by adding new features, for example, by developing new strategies in the adjudicator. However, the more complexity added into the adjudicator component, the more overhead is introduced into the execution of the N-Version components.

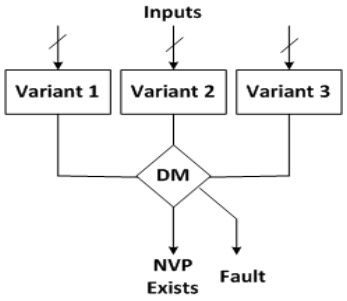


Figure 6. Example of N-Version component.

By using N-Version components, not only is fault-tolerance achieved, but it also supposes another advantage since the diagnosis stage is rendered totally unnecessary. For example, if one of the variants fails return a response in time, the adjudicator takes the results from the other variants. In consequence, diagnosis can be eliminated from the framework by using this mechanism, although it could result in a very high cost in developing and performance.

4) **Checkpointing approach**
The checkpoint mechanism is based on the idea of saving the state of the system, and, in the case of fault detection, recovering the execution of the system from the checkpoint where the state was saved. We propose the simulation of a checkpoint approach in services, whereby a recovery mechanism is launched only in the case of faults. The fault tolerance approach mechanism is composed of two parts:
- Sensors (Checkpoints). An integrity sensor is modelled as a CSP. Sensors receive data information about data inputs and outputs from the services, with which the CSP is then defined. CSP resolutions help to identify and isolate the services which are failing in run-time.
- Compensation handlers (Rollback). These are specific elements of business processes which allow the limitation of the effects created by a process when faults or errors occur. Compensation handlers allow the process execution to be rolled back from a specific point, thereby executing a set of tasks to undo the transactions already initiated. Compensation handlers are explained in next section.

The checkpoint approach presents some drawbacks in comparison with the other approaches: it requires the introduction of extra elements (sensors) into the business process design, extra time to check each sensor, and recovery of business process services in rollback. In fact the correct and minimal localization of sensors inside a business process could be a very highly complex task [35].

For high dependability, the checkpointing solution is not suitable since for long business processes, the rollback mechanism could introduce a very high overhead in the case of a fault. However, if very high dependability in our business processes is needed, then a solution with binder backup is the best solution. Nevertheless, if very high dependability is not needed, then a binder approach alone could be sufficient. If the business process with a very high level of correctness in outputs is needed, then NVP is the best solution. Although an evaluation of the number of replicas needed must be carried out since with a specific number of replicas a very high level of correctness could be ensured despite of the introduction of very high load on the development and in the adjudicator logic.

Table II provides a summarization of some characteristics of fault tolerance mechanisms applied:

- Type. The kind of fault tolerance used. For example, the replication of services in the case of a binder.
- Model-Based Diagnosis. Whether the diagnosis is necessary or not for this technique. For example, in NVP solution the diagnosis stage is unnecessary.
- Overhead. It indicates the additional logic introduced for the development of this technique. For example, in the case of binder, a dynamic binding additional logic is necessary.

IV. ILLUSTRATIVE EXAMPLE OF APPLICATION

In order to clarify the various alternatives, an example is developed. Although it is a small example, it can perfectly illustrate the problematic under consideration. In this process, there is a set of services, $S = \{S1, S2, S3, S4, S5, M1, M2, M3\}$, a set of inputs, $I = \{a, b, c, d, e, f\}$, and a set of outputs, $O = \{g, h, i\}$. The system is made up of three services *(M1, M2, M3)* with the same functionality *M(x)* but they are independent; and *(S1, S2, S3, S4, S5)* are another five elements with the same functionality *S(x)* but also independent, see Figure 7.

To illustrate a real example, the process has been distributed. The global process has been divided into three separated processes which are deployed in three different systems. The new distributed process is shown in Figure 8. In this case, there is a global process; *"Complete Process"*,

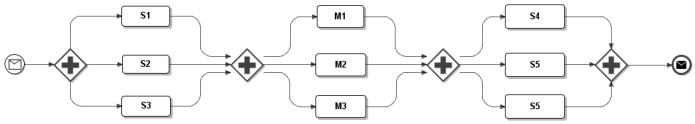


Figure 7. Example of business process.

which orchestrates the other three BPEL processes. BPEL Process 1 contains the invocations to the services S1, S2 and S3. BPEL Process 2 contains the invocations to the services M1, M2 and M3 and relies on BPEL Process 1. BPEL Process 3 contains the invocation to the services S4 and S5 and relies on BPEL Process 2 and BPEL Process 1.

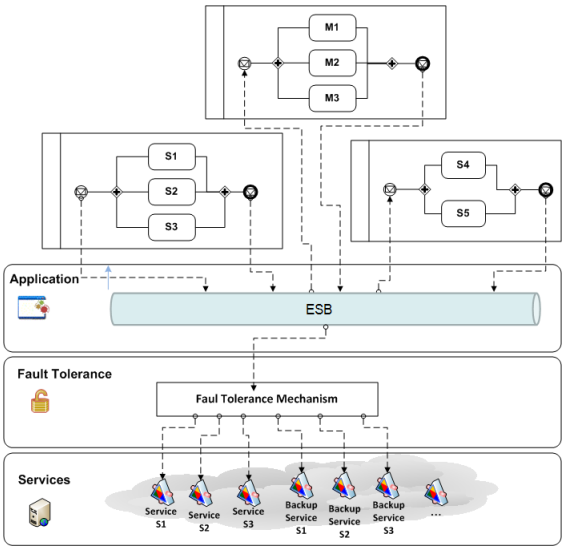


Figure 8. Distributed process.

In the scenario, it is possible to observe numerous aspects. For instance, a fault within service *S1* has a direct effect on the BPEL Process 1 result, and as a consequence the BPEL Process 2 is also affected, and finally BPEL Process 3 will be affected, and hence the final result of the complete process will be not correct. However with the correct fault tolerance mechanism this will not be the case. Therefore, fault tolerance mechanisms are necessary in order to improve the fault tolerance in the execution a business process.

1) **Primary-Backup Services with unique binder approach.** This solution combines the concept of redundant services with dynamic binding mechanisms. This approximation has a primary service as principal and one or more backups. In the case of a fault diagnosis of a service, it will be possible to replace the invocations from this primary with the backup service. A binder component is introduced between the process and the services. In Figure 9 only some services (S1, S2, and S3) have been represented, but every service has to be invoked for the binder component. The

Table II
COMPARATIVE OF THE FAULT TOLERANCE TECHNIQUES.

	Type	Model-Based Diagnosis	Overhead	Comment
Without Fault Tolerance	*NA*	+	Cost repair and detection	-
Binder	Replication (Primary/Backup)	+	Dynamic binding	Binder component
Redundant binder	Replication (Primary/Backup)	+	Dynamic binding	Replication of binder on external machine of primary
N-Version Programming	Diversity (N-Version Component)	-	Adjudicator (Voting system) & N-Replicas	Developing N-Version Components
Checkpointing	Replication (Primary/Backup)	+	Sensors & RollBack Mechanism	RollBack simulation using Compensation Handlers

binder component is common to every BPEL process. The solution is fault-tolerant although it introduces a drawback since a unique point of fault is located in the binder component. Thus, if a binder fails then all invocations to services will not be produced and all business process executions will not be able to finish.

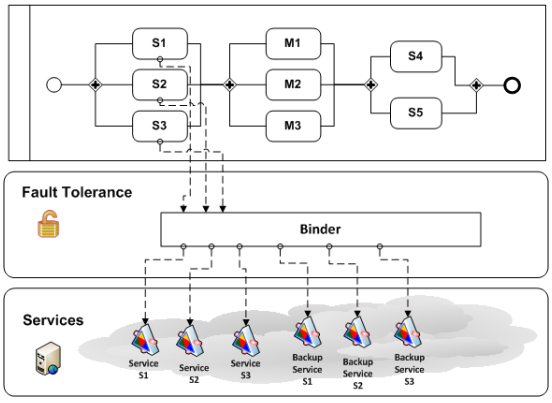


Figure 9. Fault-Tolerant solution with binder.

2) **Primary-Backup Services with replicated binder approach.** In this case a replication of the binder is introduced. If the binder component enters a fault state, then backup replica can take control of the execution, see Figure 10. In the developed proposal, the binder backup is an exact copy of the original. The binder component is common to every BPEL process.
3) **N-Version components approach.** The N-Version components use three variants with an adjudicator to obtain the output results (see Figure 11). The logic within each adjudicator is basic so that no introduce high overhead into the final performance of the process. The adjudicator therefore only compares the outputs from the variants by means of a voting system. In this implementation, a fault-tolerant software component is achieved and the diagnosis stage is rendered totally redundant.

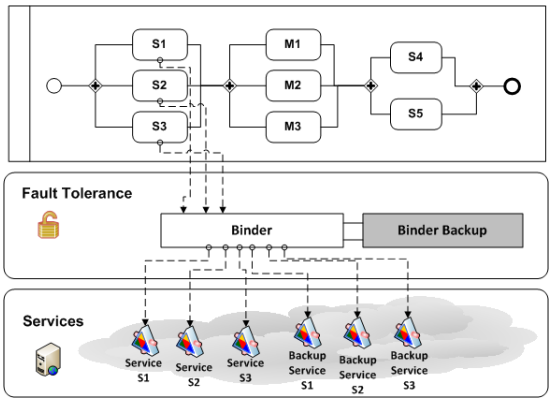


Figure 10. Fault-Tolerant solution with replicated binder.

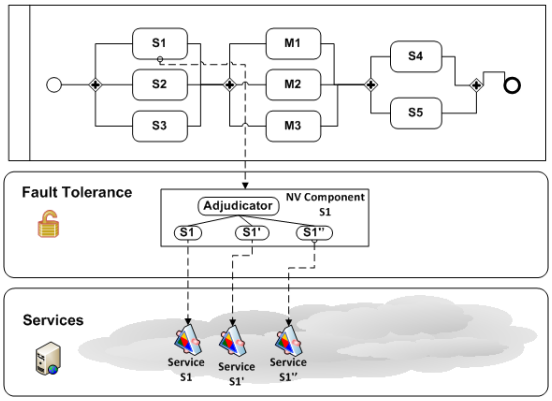


Figure 11. Fault-Tolerant solution with N-Version components.

4) **Checkpointing approach.** It is supposed that the sensors have been correctly located. Sensors, by means of a CSP resolution, indicate whether a service is

behaving as expected. In a checkpointing approach, the process state is saved at this checkpoint, but in our approach this is not necessary. Likewise, sensors provide certain information in order to help the model-based diagnosis stage determine a fault in a finite set of activities of the business process. In Figure 12, sensor IS1 covers the services S1,S2 and S3, while IS2 covers the services from IS1 and IS2.

In order to explain the functionality of a compensation handler, the following example is used: a bank has a business process which takes the data from the client so that a transaction from the client's account to his credit card can be carried out which increases the credit of the card. The service should take a specific client's data and return the amount of an account, but due to fault this service returns the amount plus other data about another account. This service is therefore not working in compliance with the specification. If this fault is determined, it could be possible to throw a fault exception and, using compensation handlers, undo all transactions carried out from the moment of the fault. Compensation handlers can only be employed when an internal fault is detected. In the case described, the services were working well but the fault came from the functionality of the service.

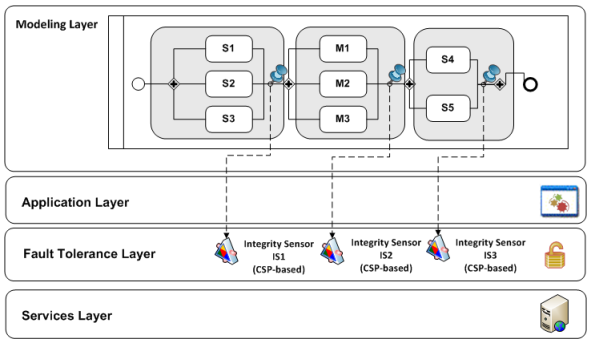


Figure 12. Approach with sensors already allocated.

Compensation is the really useful mechanism to undo transactions, but here it will be applied in another sense. In our case, when an sensor determines any fault in the execution of the business process, an exception is thrown at the end of that process. After throwing the fault, it is determined by the process and the compensation handler is invoked. Within of the compensation, the services with failures are then re-executed from a backup those correct functionality has already been tested. This process is shown in the Figure 13, where the faulty activities has been marked with a red cross within the activities. One consideration has to be taken into account, for example if the service M1 has a fault and the service S5 waits for any data generated from M1, there is a dependency between M1 and S5, therefore when M1 has a fault, then S5 has to be re-executed. In the scope of fault tolerance this solution is not purely a solution based on checkpointing since the state of the process is not saved and the execution continues from the checkpoint. The checkpointing approach does not use the same process because diagnosis is distributed from various sensors.

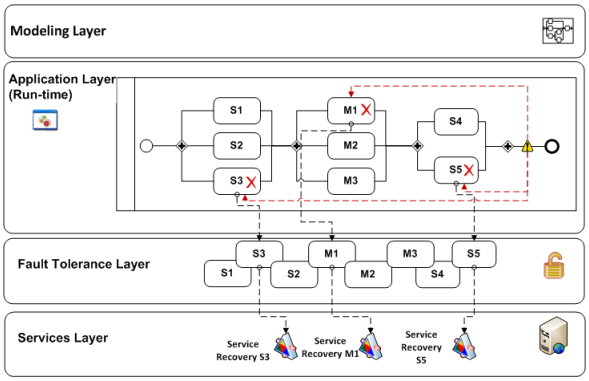


Figure 13. Recovery business process.

How does the framework work? Firstly, when the framework obtains an output, it is checked by consulting the oracle. If the oracle returns KO as response, then the framework attempt to isolate those components involved in the fault. To isolate faulty services, model-based diagnosis is employed using CSPs. In order to automate the process of diagnosis, a CSP model is solver together with the business process output. Once diagnosis retrieves which components are producing faults, then this information is translated to the fault tolerance layer and the corresponding mechanism is activated to replace erroneous services with other correct services. This process is outline depicted in Figure 14, and it is common to binder, binder backup and to NVP.

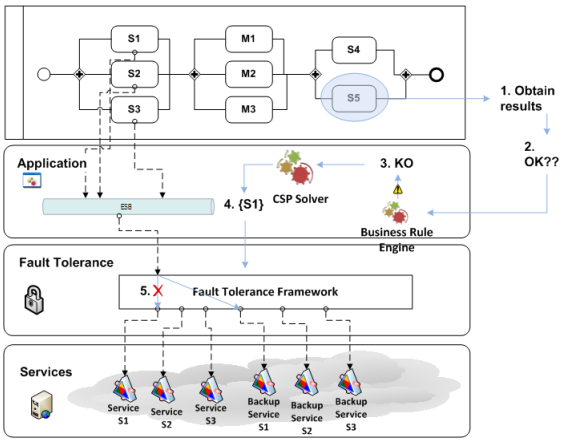
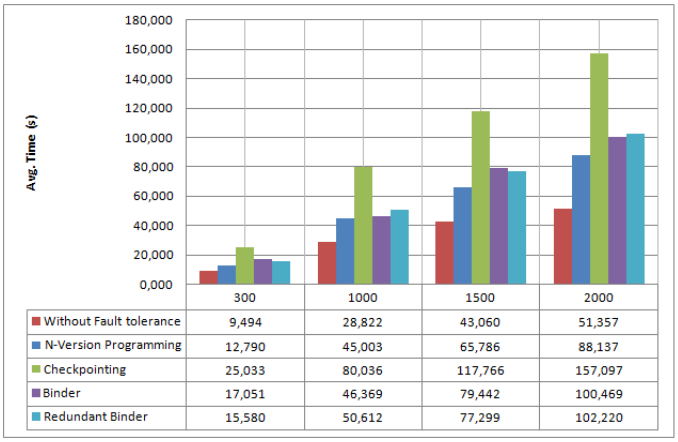
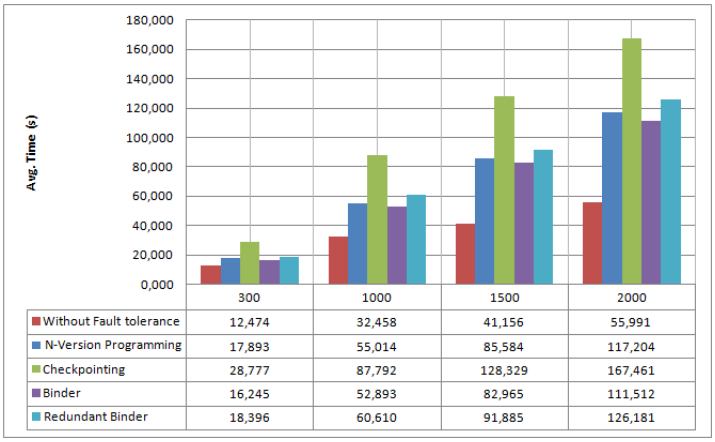


Figure 14. Example of the framework execution execution.



	300	1000	1500	2000
Without Fault tolerance	9,494	28,822	43,060	51,357
N-Version Programming	12,790	45,003	65,786	88,137
Checkpointing	25,033	80,036	117,766	157,097
Binder	17,051	46,369	79,442	100,469
Redundant Binder	15,580	50,612	77,299	102,220



	300	1000	1500	2000
Without Fault tolerance	12,474	32,458	41,156	55,991
N-Version Programming	17,893	55,014	85,584	117,204
Checkpointing	28,777	87,792	128,329	167,461
Binder	16,245	52,893	82,965	111,512
Redundant Binder	18,396	60,610	91,885	126,181

Figure 15. I. Performance one thread for execution. II. Performance more than one thread for execution.

V. EVALUATION AND RESULTS

A set of test cases has been executed for each case of the fault-tolerant approach, as described in previous sections. The tests developed have been separated in two groups: first considering one thread of execution, and second one considering more than one thread of execution. Also on, for each thread a number of invocations 300,1000,1500 and 2000 are carried out. The tests simulate the idea of there be an integrity fault in a single component for each request, for instance *S1*. For each input of the process, a set of random value tests has been created. Although many parameters could be useful in order to measure dependability properties, in our proposal the most interesting in the comparison is the performance time. Performance give a clear indication about the difference between one solution and another to be measured in terms of deliver. Figure 15 shows two graphics which contain results about average performance time . The hardware used in the execution of the tests is a server Intel Xeon E5530 2.4 GHz, with 8GB RAM and a Debian Gnu/Linux 64bits OS and a client Intel Core 2 Duo T9300 2,5GHz with 4 GB RAM.

Taking the performance parameter into account, the binder solution obtains the best result (Figure 15), although, in this case, the performance could be affected by the diagnosis stage time. In order to avoid the utilization of a binder solution, we can opt for a checkpointing approach, but would then need to value not only the time required to locate sensors, but also the overhead for checking and extra work designing handlers. On the other hand, in order to avoid the diagnosis stage, we might opt for an NVP solution. In the development sense, time spent on developing correct components using NVP must be balanced against or a solution with binder using only a primary-backup solution. In the case of multiple faults, the binder solution may be insufficient for the replication (primary and backup) of services to ensure the correction of services. However, with NVP components and the correct number of replicas we may achieve a result with a very high level of correctness.

VI. RELATED WORK

Dependability is studied in the context of business process management in [36], and a framework entitled *Dynamo* is presented. This framework provides a run-time business process supervisor that guarantees that the requirements of dependability are satisfied. The main contribution is the definition of two languages, WSCoL and WSRS, although they are not a supported standard. Likewise, [36] presents some remedial strategies that are mainly focused on the recovery context, but fail to pay attention to the typical solutions in the fault tolerance scope.

In the scope of fault tolerance for BPEL processes and Web Service composition, there are many contributions [37][38][39][40]. The feasibility of BPEL processes to implement fault tolerance techniques with BPEL language is studied in [37]. The work presents a tool for mapping fault tolerance techniques using BPEL language concepts and elements but it fails to show any example of an application or data tests with real conclusions for the work. Middleware to integrate some remedial strategies to handle violation constraint faults in the BPEL processes was developed in [39]. Whereby a framework structured in various components is developed. The most relevant components are: composition, analysis and instrumentation. The composition composes services and “business goals”, the analysis form the business process with a remedial strategy using remedial databases, and the last component, that of instrumentation, translates the process into a final BPEL process. Another studies is focused on the dynamic selection of Web Services for the construction of optimal workflows, [38]. The selection of the optimal service is based on searching from services from various repositories and data stored in databases. Although, this technique appears to provide a fault tolerance solution, this is solely due to the workflows being built on the fly

through the selection of the best service each time, but it fails to take into account the unique point of fault in the proxy component. However, we have provided a solution with a binder, and for the case of a faulty binder, we have developed another solution with a redundant binder. An architecture for self-healing BPEL processes is presented in [40]. Self-healing properties imply the development of many elements integrated into the same engine such as a monitor, a diagnoser, a planner of changes, and validation mechanisms. Some recovery strategies have been adopted in this work and integrated into the engine. Our work is more focused on mechanisms for fault-tolerance solutions, without using monitors or planners. In this sense, we have adopted an innovatory solution based on business rules (such as oracle) and CSP-based fault diagnosis.

There are some initiatives into introduction of fault-tolerance techniques in the area of Web Service, [41][42]. In these studies, the main contributions are the definition of a framework or middleware to achieve fault-tolerant service platforms. A fault-tolerance architecture for SOAP protocol is proposed and the solution is compared against the flexibility of CORBA solutions, [41]. In [42], the authors have defined and developed a mechanism to improve resilience to faults for Web service clusters to enhance the reliability of the services.

The majority of fault tolerance solutions are based on replication and recovery techniques. Although the replication is a very important concept in fault-tolerant systems, when it is necessary to create fault-tolerance in software, the solution of replication is insufficient. The philosophy in fault tolerance software is totally different; the techniques are based mainly on software diversity and data diversity, [18][34].

In fault tolerance of distributed systems, checkpointing and rollback recovery approaches are popular [43][31][44]. A well-designed checkpointing algorithm allows a faulty business process to recover the recently saved state. Further proposals have been developed in other domains such as grid computing [45] and Web Services [36].

VII. CONCLUSION AND FUTURE WORK

In this paper, an innovative framework for the development of business processes with dependable capabilities based on fault tolerance has been introduced. The framework is composed of three main elements: business rules, model-based diagnosis based on constraint programming, and fault-tolerant mechanisms. The innovation in this framework is the use of a business rule engine as an oracle of solutions, and model-based diagnosis to automate the determination and isolation of components using CSP techniques in the case of a fault. In the sense of fault tolerance, the framework presents various solutions: the first solution create fault-tolerant operational business processes (BPEL) using dynamic binding techniques and replication of services; the second solution presents an improvement introducing replication of the binder; a third solution uses the concept of software tolerance and implements NVP components; and a fourth solution provide a simulation of a checkpoint approach. To the best of our knowledge, this work is the first contribution with fault tolerance based on software fault-tolerance and checkpointing approaches for business processes.

As future work, it could be interesting to add new features of fault tolerance within the framework. The work will be extended to include with other software fault tolerance techniques such as as Recovery Block, or to introduce new features into the N-Version components, for example, changing the complexity of the adjudicator or studying the number of variations in function in terms of the service. Likewise, the framework could be extended to embrace other capabilities to achieve self-healing and self-adaptability approaches.

ACKNOWLEDGEMENTS

This work has been partially funded by the Department of Innovation, Science and Enterprise of the Regional Government of Andalusia project under grant P08-TIC-04095, by the Spanish Ministry of Science and Education project under grant TIN2009-13714, and by FEDER (under the ERDF Program).

REFERENCES

[1] A. J. Varela-Vaca, R. M. Gasca, D. Borrego, and S. Pozo, "Towards dependable business processes with fault-tolerance approach," in *Proceedings of the 2010 Third International Conference on Dependability*, ser. DEPEND 2010. IEEE Computer Society, 2010, pp. 104–111.

[2] A. J. Varela-Vaca and R. M. Gasca, "Opbus: Fault tolerance against integrity attacks in business processes," in *Computational Intelligence in Security for Information Systems 2010*, ser. Advances in Intelligent and Soft Computing, vol. 85. Springer Berlin / Heidelberg, 2010, pp. 213–222.

[3] Gartner Inc. Report, "Gartner EXP worldwide survey of nearly 1.600 CIOs shows IT budgets in 2010 to be at 2005 levels," 2010. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1283413

[4] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[5] W. M. P. van der Aalst, A. ter Hofstede, and M. Weske, "Business process management: A survey," in *Proceedings of the 1st International Conference on Business Process Management*, ser. Lecture Notes in Computer Science, vol. 2678. Springer Berlin / Heidelberg, 2003, pp. 1019–1031.

[6] A. J. Varela-Vaca, R. M. Gasca, and L. Parody, "OPBUS: Automating Structural Fault Diagnosis for Graphical Models in the Design of Business Processes," in *21th International Workshop in Principles of Diagnosis (DX'10)*, 2010, pp. 337–341.

[7] S.-M. Huang, Y.-T. Chu, S.-H. Li, and D. C. Yen, "Enhancing conflict detecting mechanism for web services composition: A business process flow model transformation approach," *Information Software Technology*, vol. 50, no. 11, pp. 1069–1087, 2008.

[8] J. Mendling, M. Moser, G. Neumann, H. M. W. Verbeek, and B. F. Vandongen, "Faulty EPCs in the SAP Reference Model," in *International Conference on Business Process Management (BPM 2006)*. Springer-Verlag, 2006, pp. 451–457.

[9] Condor Team, "Condor project," 2010. [Online]. Available: http://www.cs.wisc.edu/condor/

[10] C. Pautasso and G. Alonso, "Flexible binding for reusable composition of web services," *Software Composition*, vol. 3628/2005, no. 1820, pp. 151–166, 2005.

[11] P. Neira, R. M. Gasca, and L. Lefèvre, "Demystifying cluster-based fault-tolerant firewalls," *IEEE Internet Computing*, vol. 13, no. 6, pp. 31–38, 2009.

[12] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Mecella, "Automatic service composition based on behavioral descriptions," *International Journal of Coorporative Information Systems*, vol. 14, no. 4, pp. 333–376, 2005.

[13] D. Karastoyanova, A. Houspanossian, M. Cilia, F. Leymann, and A. Buchmann, "Extending BPEL for run time adaptability," in *Ninth IEEE International EDOC Enterprise Computing Conference EDOC05*. IEEE Computer Society, 2005, pp. 15–26.

[14] J. De Kleer and J. Kurien, "Fundamentals of model-based diagnosis," *Fault detection supervision and safety of technical processes*, pp. 25–36, 2004.

[15] A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transaction on Dependable and Secure Computating*, vol. 1, no. 1, pp. 11–33, 2004.

[16] L. Liu, Z. Wu, Z. Ma, and Y. Cai, "A dynamic fault tolerant algorithm based on active replication," in *7th International Conference on Grid and Cooperative Computing (GCC'08)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 557–562.

[17] R. Baldoni, C. Marchetti, and S. T. Piergiovanni, "Asynchronous active replication in three-tier distributed systems," in *Proceedings 9th IEEE Pacific Rim Symposium on Dependable Computing (PRDCF'02)*, 2002.

[18] L. L. Pullum, *Software fault tolerance techniques and implementation*. Norwood, MA, USA: Artech House, Inc., 2001.

[19] Object Management Group (OMG), "Business process model and notation," 2009. [Online]. Available: http://www.omg.org/spec/BPMN/1.2

[20] T. Debevoise, *Business Process Management with a Business Rules Approach: Implementing the Service Oriented Architecture*. Business Knowledge Architects, 2005.

[21] W. M. van der Aalst and A. H. M. Ter Hofstede, "Verification of workflow task structures: A petri-net-based approach," *Information Systems*, vol. 25, pp. 43–69, 2000.

[22] W. Sadiq, Maria, and E. Orlowska, "Analyzing process models using graph reduction techniques," *Information Systems*, vol. 25, pp. 117–134, 2000.

[23] H. Lin, Z. Zhao, H. Li, and Z. Chen, "A novel graph reduction algorithm to identify structural conflicts," in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, vol. 9. Washington, DC, USA: IEEE Computer Society, 2002, pp. 289–299.

[24] OASIS, "Business Process Execution Language," 2008. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[25] C. Ouyang, W. M. P. Van Der Aalst, and M. Dumas, "Translating BPMN to BPEL," *Business*, vol. 2006, pp. 1–22.

[26] S. A. White, "Using bpmn to model a bpel process," Tech. Rep., 2006.

[27] D. Chappell, *Enterprise Service Bus*. O'Reilly Media, Inc., 2004.

[28] CHOCO Team, "Choco: an open source java constraint programming library," 2010. [Online]. Available: http://www.emn.fr/z-info/choco-solver/pdf/choco-presentation.pdf

[29] D. Borrego, R. Gasca, M. Gomez, and I. Barba, "Choreography analysis for diagnosing faulty activities in business-to-business collaboration," in *20th International Workshop on Principles of Diagnosis. DX-09*, Stockholm, Suecia, 2009, pp. 171–178.

[30] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*. Elsevier, 2006.

[31] J. L. Kim and T. Park, "An efficient protocol for checkpointing recovery in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 8, pp. 955–960, 1993.

[32] A. Erradi and P. Maheshwari, "Dynamic binding framework for adaptive web services," in *ICIW '08: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 162–167.

[33] U. Küster and B. König-Ries, "Dynamic binding for BPEL processes - a lightweight approach to integrate semantics into web services," in *Second International Workshop on Engineering Service-Oriented Applications: Design and Composition (WESOA06) at 4th International Conference on Service Oriented Computing (ICSOC06)*, Chicago, Illinois, USA, 2006, pp. 116–127.

[34] W. Torres-Pomales, "Software fault tolerance: A tutorial," Tech. Rep., 2000.

[35] D. Borrego, M. T. Gomez-Lopez, R. M. Gasca, and R. Ceballos, "Determination of an optimal test points allocation for business process analysis," in *2010 IEEE/IFIP Network Operations and Management Symposium Workshops (BDIM 2010)*, 2010, pp. 159–160.

[36] L. Baresi, S. Guinea, and M. Plebani, "Business process monitoring for dependability," in *Proceedings of the Workshops on Software Architectures for Dependable Systems (WADS'06)*, 2006, pp. 337–361.

[37] G. Dobson, "Using ws-bpel to implement software fault tolerance for web services," in *EUROMICRO-SEAA*, 2006, pp. 126–133.

[38] L. Huang, D. W. Walker, O. F. Rana, and Y. Huang, "Dynamic workflow management using performance data," in *IEEE International Symposium on Cluster, Cloud, and Grid Computing*, 2006, pp. 154–157.

[39] M. Wang, K. Y. Bandara, and C. Pahl, "Integrated constraint violation handling for dynamic service composition," in *SCC '09: Proceedings of the 2009 IEEE International Conference on Services Computing*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 168–175.

[40] S. Modafferi, E. Mussi, and B. Pernici, "Sh-bpel: a self-healing plug-in for ws-bpel engines," in *MW4SOC '06: Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006)*. New York, NY, USA: ACM, 2006, pp. 48–53.

[41] C.-L. Fang, D. Liang, F. Lin, and C.-C. Lin, "Fault tolerant web services," *J. Syst. Archit.*, vol. 53, no. 1, pp. 21–38, 2007.

[42] M.-Y. Luo and C.-S. Yang, "Enabling fault resilience for web services," *Computer Communications*, vol. 25, no. 3, pp. 198 – 209, 2002.

[43] G. Cao and M. Singhal, "Checkpointing with mutable checkpoints," *Theoretical Computer Science*, vol. 290, no. 2, pp. 1127–1148, 2003.

[44] R. Baldoni, "A communication-induced checkpointing protocol that ensures rollback-dependency trackability," in *FTCS '97: Proceedings of the 27th International Symposium on Fault-Tolerant Computing (FTCS '97)*. Washington, DC, USA: IEEE Computer Society, 1997, pp. 68–77.

[45] X. Shi, J.-L. Pazat, E. Rodriguez, H. Jin, and H. Jiang, "Adapting grid applications to safety using fault-tolerant methods: Design, implementation and evaluations," *Future Generation Computer Systems*, vol. 26, no. 2, pp. 236 – 244, 2010.

Putting Theory into Practice: The Results of a Practical Implementation of the Secure Development Life Cycle

Cynthia Y. Lester
Department of Computer Science
Tuskegee University
Tuskegee, Alabama, USA
cylester@tuskegee.edu

***Abstract* – Software engineering is defined as a discipline concerned with all aspects of software production from inception to the evolution of a system. It has often been referred to as the "cradle-to-grave" approach to producing reliable, cost-efficient software delivered in a timely manner that satisfies the customer's needs. However, with the introduction of the Internet and the World Wide Web, software engineering has been required to make changes in the way that new software products are developed and protected. In order to protect systems from hackers and saboteurs in a global society where e-commerce, e-business, and e-sharing are the "norm", professionals should have sound knowledge in methods to protect data. Consequently, the area of information assurance (IA) has become one of great significance and it is important that the next generation of technologists are trained in development techniques that can ensure the confidentially and integrity of information. Traditionally, courses in secure software development are offered at the graduate level or in a stand-alone software security course at the undergraduate level. The aim of this paper is to present a framework for introducing software security to undergraduate students in a traditionally taught software engineering course. The paper focuses on and presents the results of a practical implementation of software security concepts learned through a service-learning project. The results from the study suggest that software security can be effectively introduced in a traditionally taught software engineering course through the implementation of a hands-on learning experience.**

Keywords – agile methods; secure software development service-learning; software development; software engineering; software security; traditional software development methodologies

I. INTRODUCTION

Securing information is not a new idea. In fact, securing data has its origins in World War II with the protection and safeguarding of data which resided on mainframes that were used to break codes [1]. However, during the early years, security was uncomplicated since the primary threats included physical theft of the system, espionage and sabotage against product resources [1]. Yet, it was not until the early 1970s that the concept of computer security was first studied. With the invention of the Advanced Research Projects Agency Network (ARPANET) by the U.S. Department of Defense in 1968 and its growing popularity in the early 1970s, the chance for misuse increased in what is now known to be the origin of the modern day Internet.

In 1990, it was reported that there were less that 50 million users of the Internet in the U.S. However, by 2008 the U.S. reported approximately 230,630,000 Internet users [2]. Therefore, it stands to reason that with more users and more advanced systems, the user population of today's technology would be more technically savvy than those user groups of yesteryear. However, the average user is now less likely to understand the systems of today as compared to the users of a decade ago. Further with the rapid pace at which new technologies are being introduced to the public, it becomes even more difficult for users to understand how to protect their systems and information from unwanted interruptions, threats and vulnerabilities.

In the *Report of the Presidential Commission on Critical Infrastructure Protection*, it was stated that "education on methods of reducing vulnerabilities and responding to attacks" and "programs for curriculum develop at the undergraduate and graduate levels" were recommended to reduce the number of vulnerabilities and malicious attacks on software systems [3]. Additionally, in the 2003 *National Strategy to Secure Cyberspace* four major actions and initiatives for awareness, education, and training were identified which included [4]:

- Foster adequate training and education programs to support the Nation's cybersecurity needs
- Promote a comprehensive national awareness program to empower all Americans -businesses, the general workforce, and the general population - to secure their own parts of cyberspace
- Promote private-sector support for well-coordinated, widely recognized professional cybersecurity certifications
- Increase the efficiency of existing federal cybersecurity training programs

Consequently, protecting data has become a topic of importance. In order to protect data from hackers and saboteurs in a global society where e-commerce, e-business, and e-sharing are the "norm", professionals should have sound knowledge in methods to protect data. Therefore, the

area of information assurance (IA) has become one of great significance.

Information assurance as defined in the CNSS Instruction Handbook No. 4009 are measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and nonrepudiation. Additionally, the measures include providing for restoration of information systems by incorporating protection, detection, and reaction capabilities [5]. In order for students to gain training in information assurance, a series of courses are often taken, which include traditional computer science courses but also courses in information security, network security, computer security, cryptography, software security, etc. However, unless an institution has an information assurance track or program, students may not have the opportunity to gain exposure to many of these concepts, especially those concepts found in a software security course.

Therefore, the aim of this paper is to present a framework for introducing students to concepts of software security in a traditionally taught software engineering course. The paper begins by presenting several conventional software development methodologies discussed in a traditionally taught software engineering course which lends to an argument for a paradigm shift. Additionally, the paper presents a project in which students were engaged during the course of the sixteen week semester which focused on the practical implementation of software security concepts. The results of the project are discussed as well as challenges and future work.

II. TRADITIONAL SOFTWARE DEVELOPMENT METHODOLOGIES

Software engineering is defined as "being concerned with all aspects of the development and evolution of complex systems where software plays a major role. It is therefore concerned with hardware development, policy and process design and system deployment as well as software engineering [6]."

The term software engineering was first proposed at the 1968 NATO Software Engineering Conference held in Garmisch, Germany. The conference discussed the impending software crisis that was a result of the introduction of new computer hardware based on integrated circuits [6]. It was noted that with the introduction of this new hardware, computer systems were becoming more complex which dictated the need for more complex software systems. However, there was no formalized process to build these systems which put the computer industry at jeopardy because systems were often unreliable, difficult to maintain, costly, and inefficient [6]. Consequently, software engineering surfaced to combat the looming software crisis.

Since its inception, there have been many methodologies that have emerged that lead to the production of a software product. The most fundamental activities that are common among all software processes include [6]:

- *Software specification* – the functionality of the system and constraints imposed on system operations are identified and detailed
- *Software design and implementation* – the software is produced according to the specifications
- *Software validation* – the software is checked to ensure that it meets its specifications and provides the level of functionality as required by the user
- *Software evolution* – the software changes to meet the changing needs of the customer

Typically, students are introduced to these activities in the undergraduate computer science curriculum through a software engineering course. This course is sometimes a survey course which exposes students to a variety of life cycle models used in industry. The course is often taught from a systems approach which places an emphasis on creating requirements and then developing a system to meet the requirements. In the traditional view of software development, requirements are seen as the contract between the organization developing the system and the organization needing the system [7].

A traditional view of software development is the waterfall method. The waterfall method was the first published software development process and forms the basis for many life cycles. It was noted as a great step forward in software development [8]. The method has stages that cascade from one to the other, giving it the "waterfall" name. Figure 1 is an example of the waterfall life cycle [9].

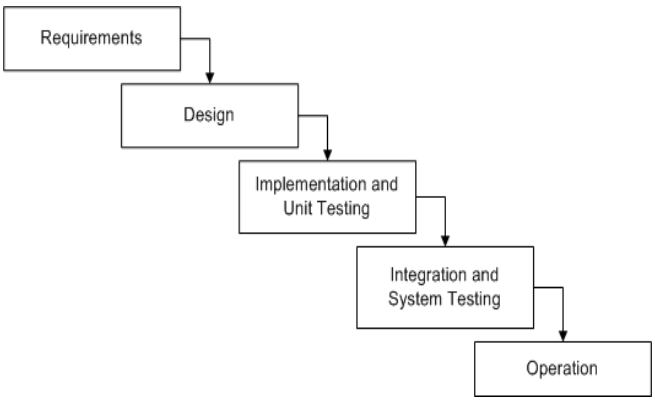


Figure 1. Waterfall model

It has been noted that the method might work satisfactorily if design requirements could be addressed prior to design creation and if the design were perfect prior to implementation [8]. Consequently, one of the main disadvantages of this model is that requirements may change accordingly to meet the needs of the customer and

the change is difficult to incorporate into the life cycle. As a result of this shortcoming, additional life cycles emerged which allowed for a more iterative approach to development.

Evolutionary development is based on the idea of developing an initial implementation and then exposing the build to the user for comment and refinement [6]. Figure 2 is an example of the evolutionary development method [6].

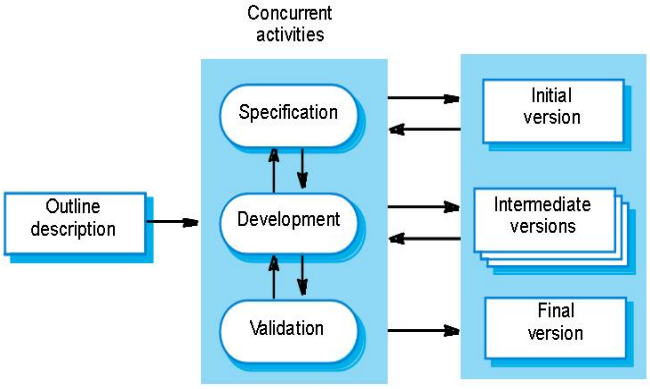


Figure 2. Evolutionary development

There are two fundamental types of evolutionary development:

- *Exploratory development* – developers work with customers to discern requirements and then the final system is delivered
- *Throwaway prototyping* – used to quickly development a concept and influence the design of the system

The advantage of evolutionary development is that it is developing specifications incrementally [6]. As customers have an opportunity to interact with the prototype, specifications are refined which leads to a better, more useful, usable, and used software. However, while this approach is somewhat better than the waterfall model, it is not without its criticisms. Sommerville notes that the process is not visible and that the systems being developed are often poorly structured [6]. The next model presented is stated to be an improvement over both the waterfall and evolutionary development models.

The spiral development model is an example of an iterative process model that represents the software process as a set of interleaved activities that allows activities to be evaluated repeatedly. The model was presented by Barry Boehm in his 1988 paper entitled *A Spiral Model of Software Development and Enhancement* [10]. The spiral model is shown in figure 3. The spiral model differs from the waterfall model in one very distinct way because it promotes prototyping; and, it differs from the waterfall and evolutionary development method because it takes into consideration that something may go wrong which is exercised through risk analysis.

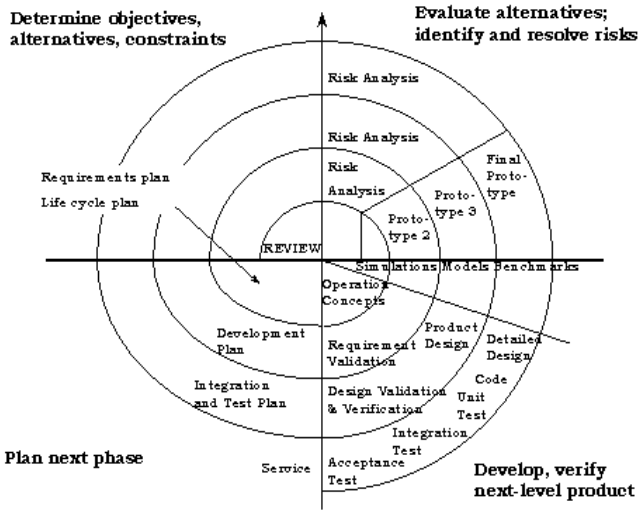


Figure 3. Spiral model

It is noted that this life cycle provides more flexibility than its more traditional predecessors. Further, this method produces a preliminary design. This phase of the life cycle was added specifically in order to identify and resolve all the possible risks in the project development. Therefore, if risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed in order to determine a possible solution.

The activities that formulate this view of software engineering came from a community that was responsible for developing large software systems that had a long life span. Moreover, the teams that used these methodologies were typically large teams with members sometimes geographically separated and working on software projects for long periods of time [7]. Therefore, software development methodologies that resulted from this view of software engineering were often termed as "heavyweight" processes because they were plan-driven and involved overhead that dominated the software process [11]. However, great difficulty occurs when these methodologies are applied to smaller-sized businesses and their systems, because these methods lack the agility needed to meet the changing needs of the user. The next section presents an overview of an emerging process methodology which is an alternative to heavyweight processes, agile development.

III. AGILE METHODS

In an effort to address the dissatisfaction that the heavyweight approaches to software engineering brought to small and medium-sized businesses and their system

development, in the 1990s a new approach was introduced termed, "agile methods." Agile processes are stated to be a family of software *development methodologies in which* software is produced in short releases and iterations, allowing for greater change to occur during the design [11]. A typical iteration or sprint is anywhere from two to four weeks, but can vary. The agile methods allow for software development teams to focus on the software rather than the design and documentation [11]. The following list is stated to depict agile methods [11], [12]:

- *Incremental design* - the design is not completed initially, but is improved upon when more knowledge is acquired throughout the process
- *User involvement* - there is a high level of involvement with the user who provides continuous feedback
- *Short releases and iterations* - allow the work to be divided, thereby releasing the software to the customer as soon as possible and as often as possible
- *Informal communication* - communication is maintained but not through formal documents
- *Minimal documentation* – source code is well documented and well-structured
- *Change* - presume that the system will evolve and find a way to work with changing requirements and environments

More specifically, the agile manifesto states:

"We are uncovering better ways of developing software by doing it and helping others to do it.
Through this work we have come to value:
Individuals and interaction over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on the right, we value the items on the left more."

It is stated that agile processes have some very precise advantages over its heavyweight predecessors which include the following as stated by Tsui and Karam [12]:

- *Low process complexity* - processes are simple which promotes easier implementation and understanding
- *Low cost and overhead* - processes require only a small number of activities that do not directly lead to the production of software
- *Efficient handling of changes* - processes are designed and developed with the presumption that requirements will change and the methodology is prepared to incorporate those changes
- *Quick results* - processes have low overhead which results in a final product being produced quicker than with traditional heavyweight processes. Also, agile processes are designed for continuous integration which allows for constant improvement and the implementation of additional functionality as the project progresses.
- *Usable systems* - the customer is involved and therefore when changes occur, the process can quickly adapt, yielding a product that the customer really wants and wants to use

However, agile methods are not without their critics. Just as the traditional methods have disadvantages, agile methods do as well. According to researchers, listed below are the main disadvantages of agile processes [11], [12]*:*

- *May not be scalable* - agile processes are typically used by small teams and may have problems scaling to adjust to larger systems without losing their agility
- *Heavy reliance on teamwork* - the processes are generally used by small teams who are centrally located and who depend on informal communication to accomplish a task; team work can be destroyed if cross-team communication mechanisms have not been designed and used
- *Reliance on frequent customer access* - it has been stated that it is sometimes difficult especially after software delivery to keep the customer involved in the process; consequently without customer involvement, agile methods may not be able to properly validate requirements or adjust to change
- *Cultural clash* – Extreme programming (XP) is probably one of the best known and most widely used agile methods [13], [14]. It was originally designed to address the needs of software development by small teams who faced changing requirements and system environments. However, XP often clashes with the more commonly accepted software engineering ideas and management techniques. Therefore, the use of agile methods by development teams may make it difficult to conduct performance evaluations and team member progress reviews.

However, just as with traditional software methodologies, agile methods do not often address software security. Moreover, when these approaches to software development are taught in traditional software engineering courses, security is mostly absent from the instruction. Hence, the increasingly important need to include a discussion of software security in the software development process taught to undergraduate students. The next section explores secure software development and its life cycle.

IV. CHARACTERISTICS OF SECURE INFORMATION

The *Morris Worm* was the first known network security breach to impact thousands of computers that were connected to (ARPANET) [15], [16]. It was reported that Robert Morris, a graduate student at Cornell University, wrote a program that exploited bugs that he noticed in several UNIX applications [16]. The basic premise of the

program was that it connected itself to another computer on the network, copied itself to the new location, and executed both the original version and the copy. This action would then be repeated in an infinite loop to other computers, thereby causing thousands of computers to become infected. He became the first person to receive a felony conviction in which he was sentenced to serve 3 years probation, 400 hours of community service, and pay a fine of $10,000 [16].

Since it began operating in 1998, the Computer Emergency Response Team (CERT) Coordination Center has tracked and reported the number of security vulnerabilities [15]. A snapshot of early security vulnerabilities is depicted in Figure 4 [15]. The chart shows the growth in security incidents from the first incident in 1998 until 1995, which was the last for which statistics were available according to the reference.

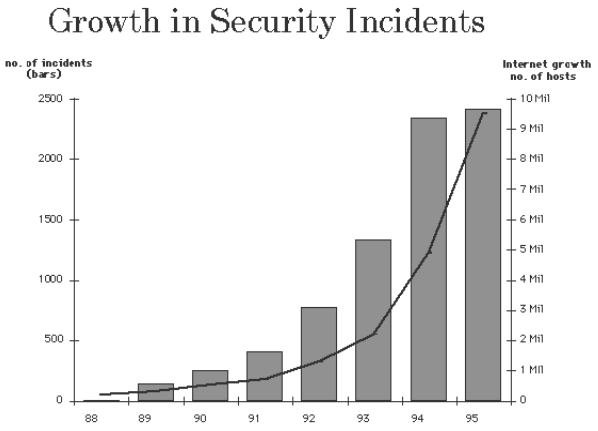


Figure 4. Vulnerabilities Report

More recently, according to statistics published by the Computer Emergency Response Team (CERT), between 1995 and 2008 approximately 44,074 vulnerabilities had been cataloged [15]. It has been reported that these software vulnerabilities and software errors cost the U.S. approximately $59.5 billion annually [17].

Software errors have grown in complexity. In 2000, NIST reported that the total sales of software reached approximately $180 billion and the software was supported by a workforce that consisted of 679,000 software engineers and 585,000 computer programmers [17]. Some of the reasons that software errors have grown in complexity are that typically, software now contains millions of lines of code, instead of thousands; the average product life expectancy has decreased requiring the workforce to meet new demands; there is limited liability among software vendors; and, there is difficulty in defining and measuring software quality [17].

Consequently, it is imperative that students in computer science and information technology be trained in the concepts of security and how to design and develop secure software so that they can contribute viably to the fast changing technological demands of this global society. The traditional development strategies expose students to the methods for software development, but as they consider how to guard against hackers, how to protect critical information, and how to lessen security threats, a question of what is "good" information arises. Therefore, before students can understand and have an appreciation for the secure software development life cycle, they must first be exposed to the qualities and characteristics of "good" information.

The value of information has been stated to come from the characteristics that it possesses [1]. While some characteristics may increase the value of the information as it relates to use by users, other characteristics may have a more significant value among security professionals. However, all characteristics as defined below are critical as it relates to secure information [1].

- *Availability* - allows users who need to access information to access the information without impediment or intrusion. Further, availability means that users can receive information in the desired format.
- *Accuracy* - as defined by *The American Heritage College Dictionary* is conformity to fact; precision; exactness [18]. As accuracy relates to secure software it means that the software has the value that the user expects and that it is also free from errors.
- *Authenticity* - is the state or quality of information being original or genuine. The information should not be a replication of other information. Whitman further reveals that information is authentic when it is the information that was originally created, placed, stored or transferred.
- *Confidentiality* - only those persons with "certain" rights can have access to the information. It means that only authorized persons or systems can gain access to the information.
- *Integrity* - is adherence to a strict code or the state of being unimpaired [1]. As it relates to the integrity of information it is the state of being uncorrupted or the state of being whole.
- *Utility* – the condition of being useful. If the information being provided is not useful or presented in a format that cannot be used, then the information loses its value or its quality of being "good" information.
- *Possession* - the condition of being owned or controlled. Whitman and Mattford note that while a breach in confidentiality always results in a breach of possession, the opposite may not be true [1].

V. THE THEORETICAL APPROACH TO THE SECURE DEVELOPMENT LIFECYCLE

There are many approaches to the development of robust software that can ensure that the information being used by users is available, accurate, authentic, possesses confidentially, integrity, is useful and can be controlled. However, the question becomes how to introduce this model at the undergraduate level when a specialized course in software security is not available or when typically students only take one course in software development. The following section presents the secure software development life cycle taught in a traditionally taught software engineering course and introduces a method which allowed students to gain practical experience in implementing security concepts.

A misconception among students as well as with computing professionals is that security should be thought of in the later phases of the software development life cycle. However, if systems are to withstand malicious attack, a robust software development model or a secure software development must be used. One viewpoint of the secure life cycle discussed in class was developed by Apvrille and Purzandi and a modified version is presented in Figure 5 [19].

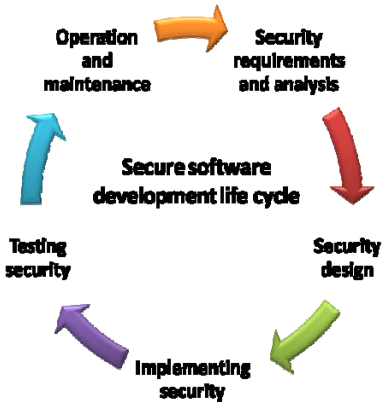


Figure 5. Secure life cycle

A. Security requirements and analysis

While requirements are being gathered from users and stakeholders, focus should also be placed on establishing a security policy. In order to develop a security policy, attention needs to be given to what needs to be protected, from whom, and for how long [20]. Additionally, thought needs to be placed on the cost of protecting the information. The result of this phase should be a set of guidelines that create a framework for security [1].

B. Security design

During the design phase it has been stated that the security technology needed to support the framework outlined in the requirements phase is evaluated, alternative solutions are explored, and a final design is agreed upon [1]. It is recommended by Viega and McGraw that the following be the focus of this phase [20]:

- How data flows between components
- Users, roles and rights that are explicitly stated or implicitly included
- The trust relationships between components
- Solutions that can be applied to any recognized problem

At the end of this phase a design should be finalized and presented. The design should be one that can be implemented.

C. Implementation

The implementation phase in the secure development life cycle is similar to that which is found in traditional methodologies. However, when implementing a software project with security in mind, it is important to consider a language or a set of languages that may have security features embedded, one that is reliable when it comes to denial-of-service attacks, and that can perform error checking statically, etc. Further, it is important to understand the weaknesses of languages, for example buffer overflows in C and C++.

D. Testing

Testing in the secure development life cycle is different than in traditional methodologies. In traditional methodologies, testing is done to ascertain the behavior of the system and to determine if the system meets the specifications. Security testing is used to determine if a system protects data and maintains functionality as intended. As mentioned previously the six concepts that need to be covered by security testing are availability, accuracy, authenticity, confidentiality, integrity, utility, and possession. It has been stated that security testing is most effective when system risks are uncovered during analysis, and more specifically during architectural-level risk analysis [20].

E. Maintenance

It has been stated that the maintenance and change phase may be the most important phase of the secure development life cycle given the high level of cleverness seen in today's threat [1]. In order to keep up with the changing threats to systems, security systems need constant updating, modifying, and testing. Constant maintenance and change ensure that systems are ready to handle and defend against threats.

VI. THE PRACTICAL APPROACH TO THE SECURE DEVELOPMENT LIFE CYCLE

A. Course Description

The course chosen for the practical implementation of the secure development life cycle was the traditionally taught *CSCI 430 – Software Engineering* course under the instruction of the author. A brief description of the course is to provide students with an engineering approach to software development and design; and, to expose students to current research topics within the field [21]. The software engineering course was modified to reinforce the need to think about security features and requirements early in the development process so that security protection mechanisms are designed and built into the system rather than added on at a later time.

The prerequisites for the course are to have successfully completed *CSCI 230 - Data Structures* and *CSCI 300 - Discrete Mathematical Structures* with a grade of C or better.

B. Course Learning Outcomes

Learning outcomes are extremely important when developing a course. The learning outcomes describe the specific knowledge and skills that students are expected to acquire. The learning outcomes for the CSCI 430 course include the following: at the end of the course, a student should be able to:

- Describe in detail the software process
- Identify various software process models and determine which model should be used for a specific project
- Implement each phase of the software process
- Work effectively and efficiently in a team environment to produce a large scale project
- Identify and discuss current research topics related to the software engineering discipline

It was the anticipation of the author that through the hands-on experience of developing a project that included security concepts, students would gain an understanding of the importance of secure software engineering and their approach to development would be enhanced. Further, as students use and understood the concepts presented during class, conceptually they would be able to apply the principles to a semester long project. It was decided to use the concepts found in service-learning to design the project. The next section provides a high level overview of service-learning.

C. Service Learning

Service-learning is defined as a method of teaching through which students apply their academic skills and knowledge to address real-life needs in their own communities [22]. Service-learning provides a compelling reason for students to learn; it teaches the skills of civic participation and develops an ethic of service and civic responsibility. By solving real problems and addressing real needs, students learn to apply classroom learning to real world situations [22]. Service-learning has been shown to be an educational technique that facilitates a student's growth in academics, communication, social maturity, critical thinking, collaboration, and leadership skills [22]. Students who are involved in meaningful service-learning have further been shown to perform better on tests, show a sense of self-esteem and purpose, connect with the community, and want to be more civically engaged than students who do not participate in service-learning activities [22].

There are many key components that are encompassed within service-learning. The author has chosen some of those activities that were included in CSCI 430 and, they are presented in the next sections.

1) Reflection. Reflection fosters the development of critical thinking in students. Reflection and critical thinking (problem-solving) are essential tools that will help students be successful in school, career, and life. Service-learning reflection includes the following activities by the student:

- Assessing personal interests, knowledge, skills, and attributes that will be useful in performing the service-learning project.
- Thinking about how to take effective steps to meet the identified needs.
- Self-evaluating one's progress toward meeting the goals of the project.

2) Working as a team. The students learn to work for a common goal and by doing so acquire a variety of skills, such as how to lead, how to be accountable, how to communicate ideas, how to listen to others, and how to set a goal and work effectively as a team to reach the goal.

3) Experiential learning. Service-learning uses direct experience and hands-on learning to help the student learn to take the initiative, assume responsibility, and develop effective problem-solving skills.

The next section describes the course project that was designed based on the concepts found in service-learning and a modified version of the secure software development life cycle.

VII. THE PROJECT

A. Project Statement

The semester long project selected for the fall 2009 semester was to develop an electronic voting/tallying system for the hotly contested position of the University's Queen. During past years, there have been errors in the selection process of the University's Queen; which has resulted in a process where contestants and the student body have little confidence. Students were required to develop a software product that meets the needs of the customer and helps to refine the election process and ballot-counting process for the University's Queen contest. Students were

part of a team which was expected to meet with the customer (or representative) so that each phase of the process could be implemented. The team was also expected to produce a deliverable by the set deadline for each phase of the process and to also deliver it and make presentations to the customer (or representative).

B. Project Learning Outcomes

The learning outcomes of the semester long project included that after the completion of the project students would:

- Have a working knowledge of the secure software development life cycle
- Understand and have a working knowledge of secure software engineering principles
- Be able to describe software vulnerabilities
- Develop and execute security measures
- Work effectively and efficiently in a team environment to produce the semester long project

C. Project Requirements

Students were given basic requirements from the instructor for the software application; however, the majority of the requirements were gathered from stakeholders. Since the project was infused with software security concepts there were both standard project requirements as well as security requirements.

D. Project Deliverables

Each item that the student team submitted was considered a deliverable. The project had four deliverables which were the requirements document, design document, implementation, and the test plan. The following is an overview of the project deliverables which were previously presented in work by Lester [23], [24].

1) Requirements Document. The first document students were required to submit was the requirements document. The requirements document was considered the official statement of what the students would implement. It included both the stakeholder requirements for the software application, which students named the *MISS System*, and a detailed specification of system requirements. To gather the requirements students met with stakeholders who included Administrators in the Office of Student Life, contestants from past elections, and student body leaders who were in charge of election results. The initial document was meant to get the students active in the planning and development of the system. After completion of the requirements document, students had an idea of the way they wanted the system to look, how the system would be accessed, and by whom (i.e., password authentication, access control).

2) Design Document. The team was required to use one of the decomposition styles discussed in the course. The design document was required to have an introduction, an overview of the design strategy chosen, and the diagrams, charts, and/or details required as part of the decomposition strategy chosen. The design document was also meant to be an in-depth description of the system design. The design showed how data flowed between system components and the trust relationships between components. Both the system and security requirements were described and explained how they would be implemented. Further the document identified vulnerabilities to the system and possible solutions were presented.

3) Implementation. Students were required to implement the project based on the requirements and design documents. To implement the project students chose the Java programming language.

4) Testing. Students were required to develop a test plan which required them to perform requirements-based testing and structural testing (inclusive of security testing).

Table 1. provides the timeframe for project deliverables.

TABLE 1. PROJECT DELIVERABLE TIME TABLE

Deliverable	**Deadline**
Requirements document	Week 8
Design document	Week 12
Implementation	Week 16
Test Plan	Week 16

VIII. RESULTS AND DISCUSSION

In order to determine the effectiveness of the service-learning project, the following actions were taken:

- For each deliverable a grade was determined based on the submitted document and the oral presentation of the document.
- After the completion of the each phase of the project, an exit interview with team members was conducted.

This section presents an overview of the results of these activities.

1) Requirements Document. The requirements documents was required to have the following sections as outlined in the textbook for the course by Sommerville [6]:

- Introduction
- User definition
- System architecture
- System models
- System evolution

Additionally, the document was graded on organization, grammar and style.

Results revealed that students had a good understanding of the user definition and the system architecture. However, system models proved to be a difficult topic for students to master. Yet, the overall quality of the document showed that students engaged in high-level critical thinking and problem solving, which was one of the goals of the service-learning project.

2) Design Document. To implement the design, students were required to choose one of the decomposition strategies discussed in class. Students chose to use object-oriented decomposition. Therefore, the parts of the document were required to include the following:

- Class diagrams
- Static diagrams (collaboration or sequence)
- Dynamic diagrams (activity or state)
- Security plan and evaluation

Results revealed that students had a good understanding of class and static diagrams, but had some difficulty with dynamic diagrams. All students had previously taken a theory course in which activity and state diagrams had been discussed, but students still struggled with the implementation of these diagrams as it related to the service-learning project.

A review of the design document also revealed that while the students gave heavy consideration and thought to security, the plan was limited in scope. The security plan addressed the characteristics of secure information, but students had difficulty with the design of the plan and how the plan would be evaluated.

3) Implementation. The requirement for this phase of the life cycle was an executable software application that met the requirements. To implement the *MISS System* students chose the Java programming language. The results from this phase of the project were mixed.

One of the challenges that students faced was the time constraint. The project was to be completed during the course of a sixteen week semester. Students naturally thought that because they had been previously engaged in semester-long projects in other courses that this project would be similar and that there would be enough time to complete all phases of the life cycle, especially the implementation phase. However, students quickly realized that this conjecture was incorrect as the end of the semester quickly approached. Further, unforeseen challenges such as changing requirements and teaming issues caused implementation delays; consequently, impacting the implementation of several requirements.

Students also had difficulty with the porting of the software application from the platform on which it was developed and the platform on which the application was to execute. The host platform was controlled by the Department of Computer Science. It was one of which the students had extensive knowledge because it was the platform on which they used for development and implementation of projects for other classes. However, the target platform on which the application was to execute was controlled by Campus Technology. It was completely different and one with which the students were quite unfamiliar. Therefore, the porting of the software application proved to the most difficult part of the project as host-target development was not considered during the requirements phase of the development life cycle.

4) Test Plan. The requirement for this phase was a test plan that included test case design. The test plan was based on Sommerville's structure of a software test plan for large and complex systems but modified to be less formal and represent the smaller nature of the *MISS System* [6]. The modified version of the software test plan included the following components:

- The testing process
- Test case design
- Hardware and software requirements
- Constraints

Test case design was an integral part of the software test plan. Test case design can be described as the process in which the system is executed in a controlled environment using established inputs into the system. The goal of the process is to create test cases that can discover defects and errors with the system and to also show that the implemented system meets the requirements of the stakeholders. The next section describes requirements-based testing and structural testing, which were used as part of the testing process.

Requirements should be designed so that they can be tested. Therefore, requirements-based testing is used to ensure that individual requirements are tested and to also provide a level of confidence to the stakeholders that their needs were being met. To test the requirements of the *MISS System* students developed and completed the following simple table as shown in Table 2.

TABLE 2. REQUIREMENT TESTING

Requirement	Test Case	Outcome

Structural testing is an approach that is used to test the system based on developer's knowledge of the structure and implementation of the software. This type of testing is typically used throughout a computer science curriculum as students who are learning to program also develop test cases based on the structure of their programs. By having knowledge of the code, student-developers can design test cases that can potentially uncover errors or problems. However, structural testing is not designed to detect missing or unimplemented requirements. Table 3 is an example of a simple table that was developed and students were asked to complete to the meet the objectives of structural testing.

TABLE 3. STRUCTURAL TESTING

Code	Test Case (Input)	Outcome

Results from testing revealed that this part of the life cycle was also quite challenging for the students. Students had some difficulty in determining test cases to test the requirements. Further, since some requirements were not

implemented, they could not be tested. Results also revealed that structural testing was a little easier for the students as this concept is one with which they are familiar because as previously stated, a modified version of structural testing is taught throughout the curriculum.

IX. CONCLUSION

In conclusion, the aim of this paper was to present a theoretical and practical framework for introducing to undergraduate students the secure software development process. The paper presents the results of a practical implementation of software security concepts learned through a service-learning project.

The author acknowledges that while there are many development methodologies that exist to train students in software security, many consist of steps that cannot be implemented in a one-semester course, especially with undergraduate students. Further, the author found that it was quite difficult for students to complete the secure development life cycle and develop a "truly" secure system, because it was costly as it related to resources (i.e., time, platform and personnel). This finding is consistent with the research perspectives of Devanbu and Stubbline [25].

Future work activities include that the author plans to revise the project, the deliverables and the timeframe for the deliverables. Additionally, the author plans to review the life cycle chosen for the project and will create a modified version of a life cycle for students to implement. The exit interviews revealed that students wanted less time for requirements/design and more time for implementation and testing.

As software becomes more complex and vulnerabilities and threats to these systems become just as complex, it is important to introduce to the next generation of technologists ways that systems can be made more secure. As educators it becomes our responsibility to train these students so that developing secure software is not just introduced in theory, but in practice as well.

ACKNOWLEDGMENTS

The author wishes to thank the students enrolled in the fall 2009 CSCI 430 – Software Engineering class for their hard work, the Tuskegee University Office of Student Life for serving as customers for the project and the Tuskegee University Office of Campus Technology for their assistance on the project.

REFERENCES

[1] M.E. Whitman and H.J. Mattford. Principles of Information Security. Boston: Course Technology. 2004.

[2] Internet users as percentage population. http://www.geohive.com/charts/ec_internet1.aspx (Accessed December 20, 2010).

[3] J. Elli, D. Fisher, T. Longstaff, L. Pesante, and R. Pethia. "A Report to the President's Commission on Critical Infrastructure Protection." [Electronic Version] http:// www.cert.org/pres_comm/cert.rpcci.ex.sum.html#edu (Accessed on April 1, 2008).

[4] The National Strategy to Secure Cyberspace. (2003). [Electronic Version]. http://www.uscert.gov/ reading_room/cyberspace_strategy.pdf (Accessed on June 13, 2011).

[5] http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf. (Accessed January 17, 2008).

[6] I. Sommerville. (2007). *Software Engineering 8th Ed*. Addison Wesley, 978-0-321-31379-9, Boston, MA.

[7] C. Angelov, R.V.N. Melnik, & J. Buur. (2003). The synergistic integration of mathematics, software engineering, and user-centered design: exploring new trends in education. *Future Generation Computer Systems*. Vol. 19, 299 – 1307.

[8] B. K. Jayaswal and P.C. Patton (2007). Design for trustworthy software: Tools, techniques for developing robust software. Prentice Hall, 0-13-187250-8, Upper Saddle Rover, NJ.

[9] Codebetter.com http://codebetter.com/blogs/raymond. lewallen/downloads/waterfalllModel.gif. (Accessed on October 10, 2009.

[10] B. Boehm. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer 21*, 5, 61-72.

[11] I. Sommerville. (2011). *Software Engineering 9th Ed*. Addison Wesley, 978-0-13-703515-1, Boston, MA.

[12] F. Tsui and O. Karam. (2011). *Essentials of Software Engineering 2nd Ed*. Jones and Bartlett Publishers, 13:978-0-7637-8634-5.

[13] K. Beck (1999). Extreme programming explained: Embrace the change. Addison Wesley.

[14] R. Jefferies, A. Anderson, C. Hendrickson. (2000). Extreme programming installed. In: The XP Series. Addison Wesley.

[15] CERT Coordination Center, CERT/CC. [Electronic Version]. http://www.cert.org/ (Accessed July 12, 2011).

[16] M. Quin. Ethics for the Information Age 4th Ed. Boston: Pearson Education. 2011.

[17] Software Errors Cost U.S. Economy $59.5 Billion Annually: NIST Assesses Technical Needs of Industry to Improve Software-Testing [Electronic Version] http://www.nist.gov/public_affairs/releases/n02-10.htm (Accessed on April 1, 2008).

[18] Accuracy; Integrity. American Heritage College Dictionary. (1993). New York: Houghton Mifflin Company.

[19] A. Apvrille and M. Purzandi. "Secure Software Development by Example," IEEE Security & Privacy, vol. 3, no. 4, July/August, 2005. p. 10 – 17.

[20] J. Viega and G. McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way.* Boston: Addison-Wesley. 2001.

[21] C. Lester. (2009). *CSCI 430 – Software Engineering Syllabus*.

[22] K. McPherson. Service Learning. New Horizons for Learning. http://www.newhorizons.org/strategies/service_learning/front_service.htm. 2005.

[23] C. Lester. (2010) "Shifting the Paradigm: Training undergraduate students in software security." Proceedings of the Fourth International Conference on Emerging Security Information, Systems and Technologies. Venice, Italy, July 18 – 25, 2011.

[24] C. Lester. (2010). "A practical application of software security in an undergraduate software engineering course." International Journal of Computer Science Issues, Vol. 7, Issue 1.

[25] P.T. Devanbu and S. Stubbline. "Software engineering for security: a roadmap," Proceedings of the Conference on the Future of Software Engineering. Limerick Ireland, June 4 – 11, 2000.

Verification of Detection Methods for Robust Human Tracking System

Hiroto Kakiuchi
System Engineering Department,
Melco Power Systems Co., Ltd.
Kobe, Japan
Email: Kakiuchi.Hiroto@zs.MitsubishiElectric.co.jp

Takao Kawamura, Toshihiko Sasama, and Kazunori Sugahara
Graduate School of Engineering,
Tottori University.
Tottori, Japan
Email: {kawamura,sasama,sugahara}@ike.tottori-u.ac.jp

***Abstract*—Much recent research is concerned with overcoming limitations of existing video surveillance systems, particularly for use in automatic human tracking systems. This paper presents detection methods which detect a lost target person during track in automatic human tracking system. The detection methods utilize an algorithm which determines the position of neighbors in a system of video cameras. By utilizing this deployed position and the view distance of video cameras, this algorithm also determines the interrelationship cameras in such a network must have in an automatic human tracking system. The system is enhanced by a video monitoring system utilizing mobile agent technologies. Mobile agents are suitable for distributed processing and parallel processing, since they can monitor their own behavior and run on distributed computers. Multiple mobile agents in the system can track numerous people using information gathered from several neighboring video cameras at the same time. Searching a target person at random is irrational when a system is losing the target; additionally, difficulty of detection can arise if the deployed position and/or the view distance of video cameras vary due to other circumstances. Therefore, a robust computation not influenced by these circumstances is needed, and detection methods utilizing the above algorithm were developed to solve these concerns and to improve reliability of this system by re-detecting the lost target.**

Keywords-Detection Method; Human Tracking; Mobile Agent.

I. INTRODUCTION

Video surveillance systems are seeing widespread use in the remote monitoring of people. Principally, the video surveillance system is used as a security system because of its ability to track a particular person. If the function of the video surveillance system is extended to track numerous people, the demands of the system are extended in various ways. Two common examples of such uses are to search for lost children and to gather/analyze consumers' route pattern for marketing research of a retail establishment. Such video surveillance systems are referred to as "automatic human tracking systems" in this paper. Our aim is to show how automatic human tracking systems can be improved by resolving some of the problems of conventional video surveillance system.

Currently, existing video surveillance systems have many limitations to their capabilities. In one case, systems have difficulty isolating a number of people located at different position at the same time and track those people automatically. In another, the number of possible targeted people is limited by the extent of users' involvement in manually switching the view from one video camera to another. Although approaches do exist to increase the efficiency of identifying and tracking particular people in a system comprised of numerous surveillance positions, these approaches demand an increase in the workload of the user since it demands users to identify the target.

Some researchers have suggested solutions to the above problems. The first approach was to use an active camera to track a person automatically [1][2], thus the camera moves in a synchronized motion along with the projected movement of the targeted person. Since a method for correcting blurring image [3] is proposed, the active camera is available. This approach is capable of locating and tracking small number of people, but improvements must be made to facilitate the locating and tracking of larger numbers of people. Another common approach was to position the camera efficiently at strategic surveillance locations [4]. This is not possible in some situations due to the number of cameras that would be necessary for full coverage, and in such cases this approach is not feasible due to limited resources. A third approach involved the implementation of sensors to efficiently track a target with multiple cameras [5][6]. A fourth approach observes a target with multiple cameras called "Watching Station" [7]. These solution also meet with resource and local restrictions such as installation barriers and the amount of area to be monitored. A fifth approach discriminates a target from color of hair, skin and clothing [8]. This solution has weak from change of hue by shade, and has possibility of mistaken recognition.

A better approach to identify and track numerous targeted people at the same time involves image processing and installation of video cameras at any designated location. However, the concern then becomes the appropriateness of using a single server when locating numerous people, since the image processing increases server load. As such, a new type of system that is capable of more efficiently identifying and locating people must be developed. In this proposed system, utilizing mobile agent technologies, the ratio of mobile agents and tracked targets is directly proportional

[9][10][11][12]. According to many studies, an agent-based approach is appropriate for distributed systems and parallel processing [13][14][15], since mobile agents can transfer copies of themselves to other servers in the system. By working cooperatively, such a multi-agent system would be effective [16]. With distributed processing, mobile agent technologies are more effective and efficient than conventional video surveillance systems, assuming that a large number of servers with video camera are installed. If one mobile agent can track one person, then multiple mobile agents can track numerous people at the same time, and the server balances the load process of the operating mobile agent on each server with a camera. A video surveillance system enhanced with mobile agent technologies is called "Automatic Human Tracking System" [17][18]. In such a system, a mobile agent tracks a person captured by a video camera and a server process the data. The video camera and the server are treated as a single entity since the video camera and the server are deployed at the surveillance position. Upon initialization of a person as a target to track, a mobile agent is generated for that particular person. After verifying the features of the person [19], the mobile agent tracks the movement of the person by utilizing the neighbor camera/server location information.

In the automatic human tracking system, tracking function must be robust even if the system loses a target person. Present image processing is not perfect because a feature extraction like SIFT [20] has high accuracy but takes much processing time. The trade-off of accuracy and processing time is required for such a feature extraction algorithm. In addition, the speed a person walks is various and the person may be unable to be captured correctly in cameras. Therefore, it is necessary to re-detect a target person as tracking function even if the system loses the target. We propose two types of detection method to detect a target person in this paper. The detection methods compensate for the above weakness of feature extraction as a function of system. The detection methods also utilize "neighbor node determination algorithm" [21] to detect the target efficiently. The algorithm can determine neighbor camera/server location information without the location and view distance of video camera. Neighbor camera/servers are called "neighbor camera node/nodes" in this paper. The mobile agent can detect the target person efficiently with knowing the neighbor camera node location information.

On the following sections, Section II will be describing about overview of automatic human tracking system, Section III contains the overview of Neighbor node determination algorithm, Section IV explains the two types of detection method to detect a target person when the system is losing the target, Section V is the results of examination using the detection methods, and Section VI is the conclusion of the detection methods and feature subjects.

II. Overview of Automatic Human Tracking System

The system configuration of the automatic human tracking system is shown in Figure 1. It is assumed that the system is installed in a given building. Before a person is granted access inside the building, the person's information is registered in the system. Through a camera an image of the persons face and body is captured. Feature information is extracted from the image by SIFT and registered into the system. Any person who is not registered or not recognized by the system is not allowed to roam inside the building. This system is composed of an agent monitoring terminal, agent management server, video recording server and feature extraction server with video camera. The agent monitoring terminal is used for registering the target person's information, retrieving and displaying the information of the initiated mobile agents, and displaying video of the target entity. The agent management server records mobile agents' tracking information history, and provides the information to the agent monitoring terminal. The video recording server records all video images and provides the images to the agent monitoring terminal via request. The feature extraction server along with the video camera analyzes the entity image and extracts the feature information from the image.

A mobile agent tracks a target entity using the feature information and the neighbor nodes information. The number of mobile agents is in direct proportion to the number of the target entities. A mobile agent is initialized at the agent monitoring terminal and launched into the feature extraction server. The mobile agent extracts the features of a captured entity and compares it with the features already stored by the agent. If the features are equivalent, the entity is located by the mobile agent.

The processing flow of the proposed system is also shown in Figure 1. (i) First, a system user selects an entity on the screen of the agent monitoring terminal, and extracts the feature information of the entity to be tracked. (ii) Next, the feature information is used to generate a mobile agent per target which is registered into the agent management server. (iii) Then the mobile agent is launched from the terminal to the first feature extraction server. (iv) When the mobile agent catches the target entity on the feature extraction server, the mobile agent transmits information such as the video camera number, the discovery time, and the mobile agent identifier to the agent management server. (v) Finally, the mobile agent deploys a copy of itself to the neighbor feature extraction servers and waits for the person to appear. If the mobile agent identifies the person, the mobile agent notifies the agent management server of the information, removes the original and other copy agents, and deploys the copy of itself to the neighbor feature extraction servers again. Continuous tracking is realized by repeating the above flow.

The system architecture is shown in Figure 2. The GUI is

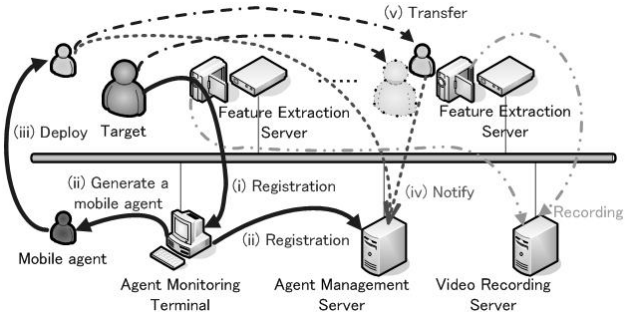


Figure 1. System Configuration and Process Flow.

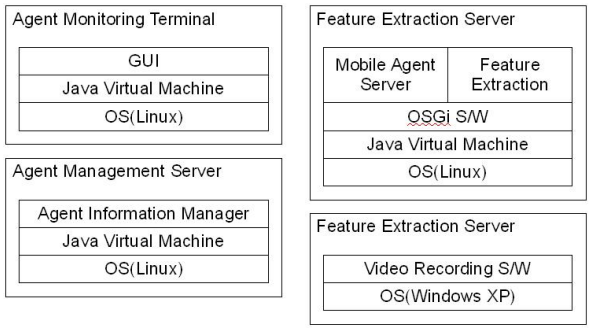


Figure 2. System Architecture.

operated only on the agent monitoring terminal. The GUI is able to register images of the entities and monitor the status of all the mobile agents. The mobile agent server is executed on the feature extraction server and allows the mobile agents to execute. The Feature extraction function is able to extract features of the captured entities, which is then utilized in the tracking of those entities as mobile agents. OSGi S/W [22] acts as a mediator for the different software, allowing the components to utilize each other. The Agent information manager manages all mobile agent information and provides the information to the agent monitoring terminal. The Video recording S/W records all video, and provides the video movie to agent monitoring terminal. Each PC is equipped with an Intel Pentium IV 2.0 GHz processor and 1 GB memory. The system has an imposed condition requirement that maximum execution time of feature judgment is 1 second and maximum execution time of mobile agent transfer is 200 milliseconds.

In this system, a simulator is also currently being developed in Java Language. The simulator consists of an image processing simulator and simulator tools. The simulator tools are an editor for the creation of target simulation routes and a simulation feature data creator. The simulator tools are shown in Figure 3 and Figure 4. Since it is difficult to place many cameras, the image processing simulator is performed on the feature extraction server instead of a genuine image processing function. In addition, this simulator changes a target entity's feature to a walking target entity by using a

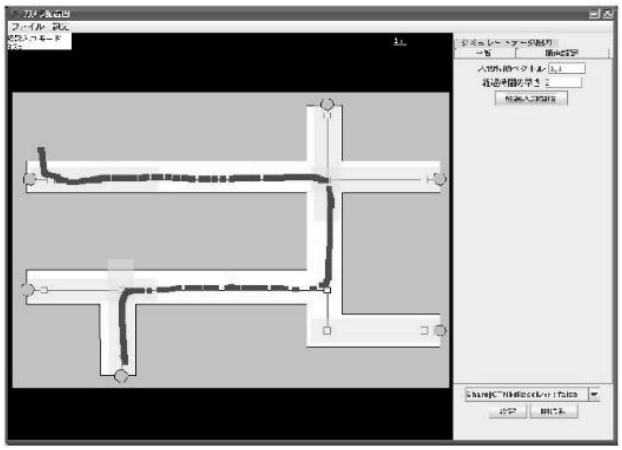


Figure 3. Editor of Simulation Route.

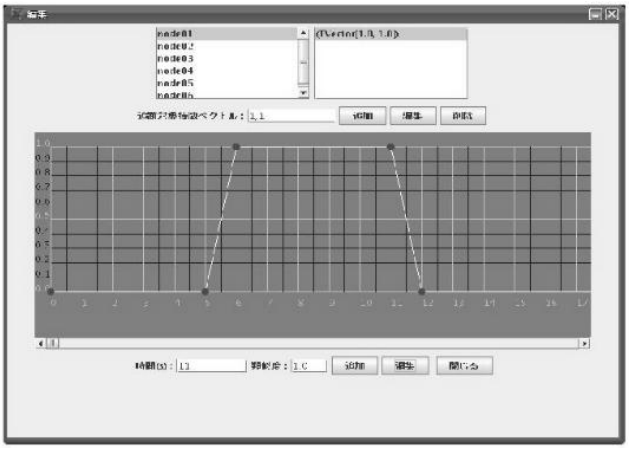


Figure 4. Creator of Simulation Feature Data.

simulation agent. The simulation agent is also a mobile agent that simulates the movement of a target entity and changes the target entity feature. The movement of the target entity is digitized by the editor of route simulation and the target entity features are digitized by the simulation feature data creator.

III. Overview of Neighbor Node Determination Algorithm

If a mobile agent tracks a target entity, the mobile agent has to know the deployed location of the video cameras in the system. However the abilities of the neighbor cameras are also determined by their view distances. A problem caused by a difference in the view distances can occur. This problem occurs when there is a difference in expected overlap of a view or an interrupt of view.

A scenario in which a neighbor video camera's location is influenced by view distance is shown in Figure 5. The upper side figures of Figure 5 show four diagrams portraying a floor plan with four video cameras each, considering

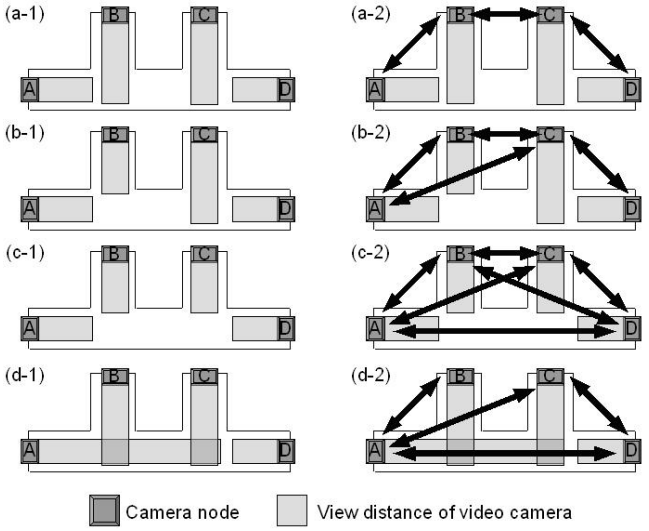


Figure 5. Influence by Change of View Distance of Video Cameras.

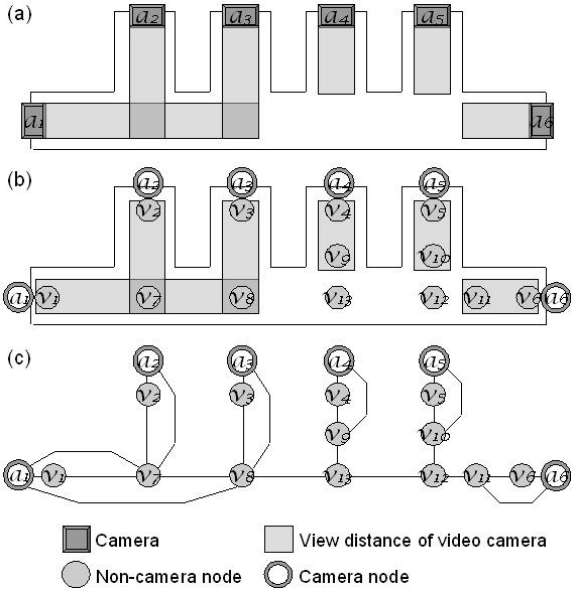


Figure 6. Figure that sets Non-camera Nodes.

the view distances of each video camera are different and assuming that the target entity to be tracked moves from the location of video camera A to video camera D. The underside figures of Figure 5 show neighbors of each video camera with arrows. The neighbor of video camera A in object a-1 of Figure 5 is video camera B but not C and not D as the arrows in object a-2 show. In object a-1 of Figure 5, video camera C and D are also not considered neighbors of video camera A, because video camera B blocks the view of video camera C and D. And the target entity can be captured at an earlier time on video camera B. But in the case of object b-1 of Figure 5, the neighbors of video camera A are video camera B and C but not camera D as the arrows in object b-2 of Figure 5 show. In the case of object c-1 of Figure 5, the neighbors of video camera A are all video cameras as the arrows in object c-2 of Figure 5 show. Thus neighbor video camera's location indicates the difference in view distances of video cameras. The case of object d-1 in Figure 5 is more complicated. The neighbors of video camera A in object d-1 of Figure 5 are video camera B, C, and D as the arrows in object d-2 of Figure 5 show. And video camera B is not considered the neighbor of video camera C. It is because video camera A exists as a neighbor between video camera B and C. When it is assumed that a target entity moves from A to D, the target entity is sure to be captured by video camera A, B, A, and C in that order.

This scenario indicates that the definition of "neighbor" cannot be determined clearly because the determination of the neighbor definition is influenced by the change of view distance and it becomes more complicated as the number of video cameras increases.

Neighbor node determination algorithm can easily determine the neighbor video camera's location without regard to the influence of view distances and any modification of the information of the currently installed cameras. The modification information is set in the system to compute neighbor video cameras on the diagram, which is expressed as a graph. Nodes are used to compute neighbor video camera's information in this algorithm. The nodes are defined as camera node and non-camera node. Camera node is the location of video camera that is labeled as camera node. The nodes are defined as $A = \{a_1, a_2, ..., a_p\}$. This node is also a server with video camera. Non-camera node is defined as $V = \{v_1, v_2, ..., v_q\}$. The conditions of a non-camera node are stated below; i) either of crossover, corner, terminal of passage, ii) the position where a video camera is installed, or iii) the end point of the view distance of a video camera. In addition, the point where the above conditions are overlapped is treated as one node. When the view distance of the video camera reaches a non-camera node, the non-camera node is defined as the neighbor of the camera node. When two non-camera nodes are next to each other on a course, those nodes are specified as neighbors. Figure 6 shows an example of these definitions applied and shows the view distances of the video cameras.

The algorithm accomplishes this using an adjacency matrix. Two kinds of adjacency matrix are used by the algorithm. One is an adjacency matrix X made from camera nodes' locations as rows and non-camera nodes' locations as columns. Element x_{ij} of matrix X is defined as (1). Table I is the adjacency matrix X which is a table representation based on the object (c) of Figure 6.

$$x_{ij} = \begin{cases} 1 & \textit{There is the line which links} \\ & \textit{camera node } a_i \textit{ and non-camera node } v_j. \\ 0 & \textit{There is no link.} \end{cases} \quad (1)$$

Table I
ADJACENCY MATRIX X WITH ELEMENT x_{ij}

X	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
a_1	1	0	0	0	0	0	1	1	0	0	0	0	0
a_2	0	1	0	0	0	0	1	0	0	0	0	0	0
a_3	0	0	1	0	0	0	0	1	0	0	0	0	0
a_4	0	0	0	1	0	0	0	0	1	0	0	0	0
a_5	0	0	0	0	1	0	0	0	0	1	0	0	0
a_6	0	0	0	0	0	1	0	0	0	0	1	0	0

Table II
ADJACENCY MATRIX Y WITH ELEMENT y_{ij}

Y	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
v_1	0	0	0	0	0	0	1	0	0	0	0	0	0
v_2	0	0	0	0	0	0	1	0	0	0	0	0	0
v_3	0	0	0	0	0	0	0	1	0	0	0	0	0
v_4	0	0	0	0	0	0	0	0	1	0	0	0	0
v_5	0	0	0	0	0	0	0	0	0	1	0	0	0
v_6	0	0	0	0	0	0	0	0	0	0	1	0	0
v_7	1	1	0	0	0	0	0	0	0	0	0	0	0
v_8	0	0	1	0	0	0	0	0	0	0	0	0	1
v_9	0	0	0	1	0	0	0	0	0	0	0	0	1
v_{10}	0	0	0	0	1	0	0	0	0	0	0	1	0
v_{11}	0	0	0	0	0	1	0	0	0	0	0	1	0
v_{12}	0	0	0	0	0	0	0	0	0	1	1	0	1
v_{13}	0	0	0	0	0	0	0	1	1	0	0	1	0

Another one is as adjacency matrix Y made from non-camera nodes' location as rows and columns. Element y_{ij} of matrix Y is defined as (2). Table II is the adjacency matrix Y which is a table representation based on the object (c) of Figure 6.

$$y_{ij} = \begin{cases} 1 & \textit{There is the line which links} \\ & \textit{two non-camera nodes, } v_i \textit{ and } v_j. \\ 0 & \textit{There is no link or (3) is satisfied.} \end{cases} \quad (2)$$

$$y_{ij} = y_{ji} = 1, \sum_{n=1}^{m} x_{ni} \geq 1, \sum_{n=1}^{m} x_{nj} \geq 1 \quad (3)$$

The neighbor information for video cameras is calculated from the connection information of non-camera nodes by using adjacency matrix X and Y.

Below is the algorithm to determine neighbor nodes: i) Set camera nodes and non-camera nodes on the diagram as shown in object (b) of Figure 6. ii) Transform the diagram to a graph as shown in object (c) of Figure 6. iii) Generate an adjacency matrix X from camera node locations and non-camera node locations on the graph, and generate an adjacency matrix Y from non-camera node locations on the graph. Adjacency matrix X indicates that rows are camera nodes and columns are non-camera nodes. Adjacency matrix Y indicates that rows and columns are non-camera nodes, which results in adjacency matrix Y resolving an overlap problem of view distances between video cameras. iv) Calculate adjacency matrix X' and Y' by excluding unnecessary non-camera nodes from adjacency matrix X and Y. v) Calculate neighbor's location matrix by multiplying adjacency matrix and transposed matrix X'^T. This neighbor's location matrix is the neighbour's node information. An unnecessary non-camera node is a non-camera node which has no camera node as a neighbor. Adjacency matrix X' and Y' are computed without unnecessary nodes, and using the procedure shown later. There are reasons why it might be better to include the unnecessary nodes in the diagram from the beginning as we have done. Since the risk of committing an error will be higher as the diagram becomes larger, we include the unnecessary nodes from the beginning and remove them at the end. Table III is the adjacency matrix X' without unnecessary nodes from the adjacency matrix X, and Table IV is the adjacency matrix Y' without unnecessary nodes from the adjacency matrix Y. Finally, matrix E which indicates the neighbor nodes is derived as (4). In addition, e_{ij} which is a value of element of E indicates number of route for reaching from camera node a_i to camera node a_j.

Table III
ADJACENCY MATRIX X'

X'	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
a_1	1	0	0	0	0	0	1	1	0	0	0
a_2	0	1	0	0	0	0	1	0	0	0	0
a_3	0	0	1	0	0	0	0	1	0	0	0
a_4	0	0	0	1	0	0	0	0	1	0	0
a_5	0	0	0	0	1	0	0	0	0	1	0
a_6	0	0	0	0	0	1	0	0	0	0	1

Table IV
ADJACENCY MATRIX Y'

Y'	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
v_1	0	0	0	0	0	0	1	0	0	0	0
v_2	0	0	0	0	0	0	1	0	0	0	0
v_3	0	0	0	0	0	0	0	1	0	0	0
v_4	0	0	0	0	0	0	0	0	1	0	0
v_5	0	0	0	0	0	0	0	0	0	1	0
v_6	0	0	0	0	0	0	0	0	0	0	1
v_7	1	1	0	0	0	0	0	0	0	0	0
v_8	0	0	1	0	0	0	0	1	1	1	1
v_9	0	0	0	1	0	0	0	1	1	1	1
v_{10}	0	0	0	0	1	0	0	1	1	1	1
v_{11}	0	0	0	0	0	1	0	1	1	1	1

$$E = X'Y'X'^T \begin{cases} \geq 1 & a_i \textit{ is neighbor node to } a_j \\ = 0 & a_i \textit{ is not neighbor node to } a_j \end{cases} \quad (4)$$

IV. DETECTION METHODS

The detection method in this paper is used to re-detect a target when the automatic tracking system loses the target.

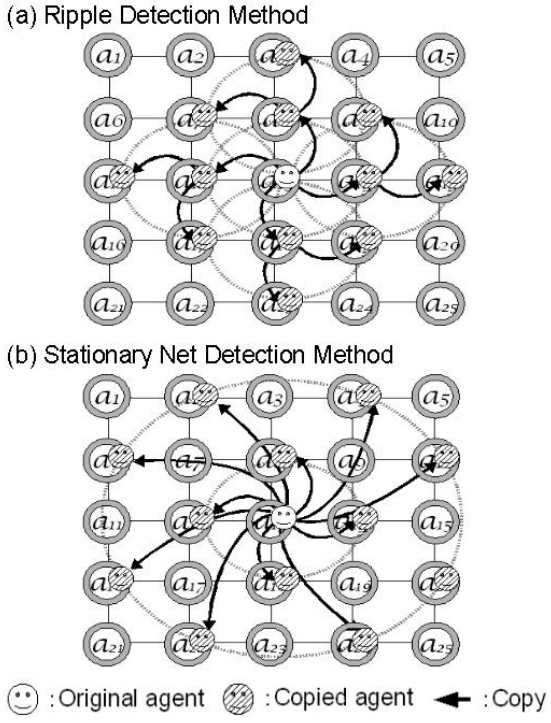


Figure 7. Detection Methods.

This method improves the tracking function, because an individual can not be accurately identified in the current image processing. As such the reliability of the system is further improved, because it enhances the continuous tracking function and re-detection of the target even if a target is lost for a long period of time. In this paper, if a target is not captured within a certain period of time, the mobile agent then concludes that the target is lost. On such case the system can also conclude that the target is lost.

We are proposing two types of detection method: (a) "Ripple detection method" and (b) "Stationary net detection method". These methods are shown in Figure 7.

Ripple detection method widens a search like a ripple from where an agent lost a target to give top priority to re-detect. This method has a feature that the discovery time becomes shorter and usual tracking can resume more quickly, if the target exists near where the agent lost. In addition, this method deletes other agents immediately after discovering the target, and suppresses the waste of the resource. The Ripple detection method is developed and is examined in search propriety. In the Ripple detection method, the neighbor camera nodes are shown as (5).

$$E1 = E = X'Y'X'^{T} \tag{5}$$

When a mobile agent lost a target, copy agents are deployed to the next nodes of (5) expressed by (6), and search is started. E^2 shows next neighbor camera nodes, because the elements of E^2 larger than 1 can be reached if the elements are larger than 1. Therefore, excepting neighbor node information E of camera nodes, automatic human tracking system uses a minimum resource by deploying copy agents.

$$E2 = E^2 - E1 = E^2 - E \tag{6}$$

Similarly, it becomes like (7) and (8) to calculate the next camera node further.

$$E3 = E^3 - (E2 + E1) = E^3 - E^2 \tag{7}$$

$$E4 = E^4 - (E3 + E2 + E1) = E^4 - E^3 \tag{8}$$

As mentioned above, the equation (9) is derived when deploying agents efficiently to the n next camera nodes. n is larger than 2 and is incremented one by one when this equation is used for detection.

$$En = E^n - \sum_{m=1}^{n-1} Em = E^n - E^{n-1} \tag{9}$$

Stationary net detection method widens a search like setting a stationary net with the Neighbor node determination algorithm from where an agent lost a target to give top priority to re-detect. This method uses (10) in the algorithm.

$$E = X'(Y')^n X'^{T} \begin{cases} \geq 1 & a_i \textit{ is neighbor node to } a_j \\ = 0 & a_i \textit{ is neighbor node to } a_j \end{cases} \tag{10}$$

In this equation, adjacency matrix E indicates the node that can reach via n non-camera nodes and n is always set to $n \geq 2$. In this method, the coefficient n is set to $n = 4$ because camera nodes are set with a certain interval. The interval between cameras in the real system may be close, but in that case, number of non-camera nodes between the cameras decreases. Therefore it is enough interval to re-detect a target if n consists of $n \geq 4$. This method has a feature that agents are deployed to neighbor camera nodes via n next non-camera nodes and catch a target like a stationary net. In addition, this method also deletes other agents immediately after discovering the target, and suppresses the waste of the resource. The Stationary net detection method is developed and is examined in search property. In the Stationary net detection method, the neighbor camera nodes are shown as (11).

$$E1 = E = X'Y'X'^{T} \tag{11}$$

When a mobile agent lost a target, copy agents are deployed to the next nodes of (11) expressed by (12), and search is started. $X'Y'^{2}X'^{T}$ shows neighbor camera nodes via two non-camera nodes, because the elements of $X'Y'^{2}X'^{T}$ larger than 1 can be reached if the elements are larger than 1. If copy agents are deployed at each camera nodes via non-camera nodes more than two, detection range of target widens. And, excepting neighbor node information E

of camera nodes, automatic human tracking system uses a minimum resource by deploying copy agents.

$$E2 = X'Y'^2X'^T - E1 \tag{12}$$

Similarly, it becomes like (13) and (14) to calculate the next camera node of more wide range.

$$E3 = X'Y'^3X'^T - (E2 + E1) \tag{13}$$

$$E4 = X'Y'^4X'^T - (E3 + E2 + E1) \tag{14}$$

As mentioned above, the equation (15) is derived when deploying agents efficiently to the next camera nodes via n non-camera nodes. n is larger than 2 and is incremented one by one when this equation is used for detection.

$$En = X'Y'^nX'^T - \sum_{m=1}^{n-1} Em = X'Y'^nX'^T - X'Y'^{n-1}X'^T \tag{15}$$

V. Examination

Examination environment for the Ripple detection method and the Stationary net detection method is shown in Figure 8 and Figure 9. There are twelve camera nodes in the environment of floor map 1, and there are fourteen camera nodes in the environment of floor map 2. Here, the following conditions are set in order to examine the effectiveness of these detection methods. i) Camera nodes are arranged on latticed floor, $56m \times 56m$. ii) View distance of camera is set to $10m$ in one direction. iii) Identification of a target in the image processing does not fail when re-detecting. iv) Walking speed of the target is constant. v) Only one target is searched. vi) The target moves only forward without going back. In the case of the floor map 1, the target moves following the order of a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12} and a_1. In the case of the floor map 2, the target moves following the order of a_1, a_2, a_4, a_5, a_7, a_9, a_{10}, a_{11} and a_1. In the examination, the time that an agent concludes a failure of tracking is same as search cycle time. The search cycle time is defined as the time concluded that an agent can not discover a target. The search cycle time is prepared using 3 patterns 12 seconds, 9 seconds and 6 seconds. Walking speed of the target is prepared using 3 patterns $1.5m/s$, $2m/s$ and $3m/s$. And search of target is prepared that an agent loses a target at a_7 and the agent starts a search in the situation that the target has already moved to a_8. Furthermore, Stationary net detection method is examined by 3 patterns $n = 2$, $n = 3$ and $n = 4$, because of confirming effectiveness by number of non-camera nodes. On each floor map, using 12 patterns of such combination by each walking speed, discovery time and the number of agents are measured. Generally, the walking speed of a person is around $2.5m/s$, and the two types of walking speed, $2m/s$ and $3m/s$, used by the target which was examined are almost equivalent to the walking speed of general person.

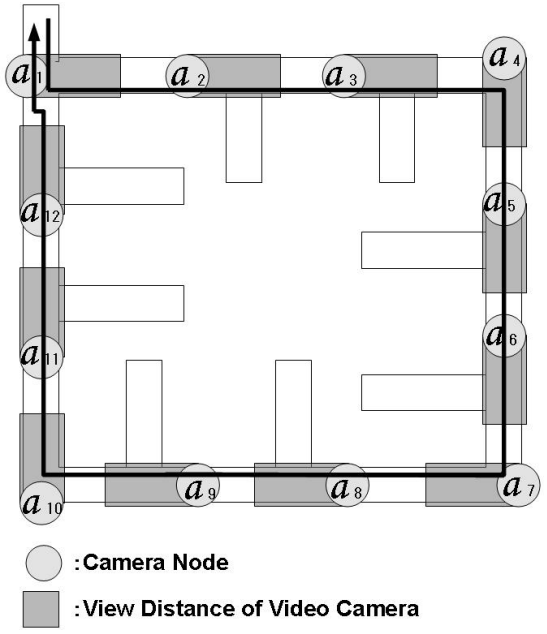


Figure 8. Floor Map 1 for Examination of Detection Methods.

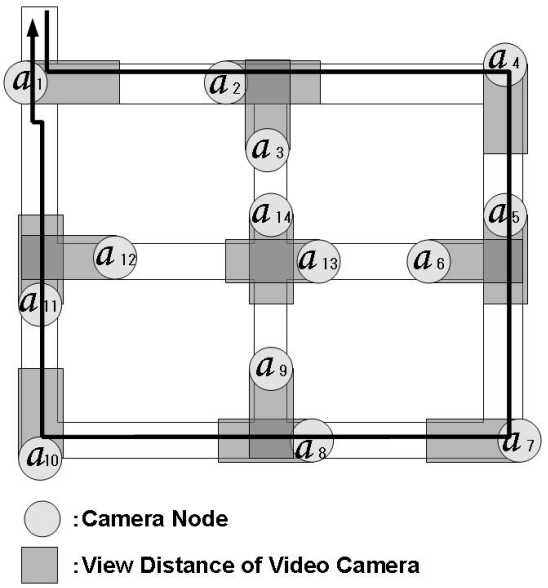


Figure 9. Floor Map 2 for Examination of Detection Methods.

And walking speed, $1.5m/s$, is very slow from the walking speed of general person.

The results of the measurement on the floor map 1 are shown in Table V, Table VI and Table VII. The results of the measurement on the floor map 2 are shown in Table VIII, Table IX and Table X. They are a mean value of 5 measurements.

The result of the Ripple detection method shows that the discovery time becomes shorter and usual tracking can resume more quickly, if the target exists near where the agent

Table V
DETECTION TIME ON FLOOR MAP 1 BY WALKING SPEED 1.5M/S

Walking Speed(1.5m/s)		Ripple	Stationary Net (n=2)	Stationary Net (n=3)	Stationary Net(n=4)
Search Cycle (12s)	Number of Agents	6	12	12	6
	Discovery Time (s)	20.6	–	–	20.5
Search Cycle (9s)	Number of Agents	6	12	6	6
	Discovery Time (s)	20.5	–	20.5	20.5
Search Cycle (6s)	Number of Agents	7	6	6	7
	Discovery Time (s)	20.5	20.5	20.5	20.5

Table VI
DETECTION TIME ON FLOOR MAP 1 BY WALKING SPEED 2M/S

Walking Speed(2m/s)		Ripple	Stationary Net (n=2)	Stationary Net (n=3)	Stationary Net(n=4)
Search Cycle (12s)	Number of Agents	6	12	12	6
	Discovery Time (s)	15.5	–	–	15.5
Search Cycle (9s)	Number of Agents	6	12	12	6
	Discovery Time (s)	15.5	–	–	15.4
Search Cycle (6s)	Number of Agents	7	12	6	7
	Discovery Time (s)	16.5	–	15.5	15.7

lost. But, if the walking speed of a target is faster, the agent will become difficult to discover the target.

The result of the Stationary net detection method shows that the agent can discover a target if coefficient n has larger value, even if the walking speed of a target is faster. And it is not enough interval to re-detect a target if n consists of $n \leq 3$ and it is not enough time to re-detect the target if the search cycle time is shorter.

From the result of measurement on the floor map 1, if the Stationary net detection method uses coefficient $n = 4$, there is not the difference of efficiency between the Ripple detection method and the Stationary net detection method. However, from the result of measurement on the floor map 2, if a floor map is complicated, the discovery time of the Stationary net detection method becomes shorter than the discovery time of the Ripple detection method and the number of agents of the Stationary net detection method becomes less than the number of agents of the Ripple detection method.

On the whole, the result of both methods shows that a number of agents decreases by searching a target near search cycle but the agents can not search the target if the search cycle time is longer than the waking speed. In addition based on the results, when the walking speed is faster, the discovery time is shortened or equal and the number of agents decreases or is equal.

Table VII
DETECTION TIME ON FLOOR MAP 1 BY WALKING SPEED 3M/S

Walking Speed(3m/s)		Ripple	Stationary Net (n=2)	Stationary Net (n=3)	Stationary Net(n=4)
Search Cycle (12s)	Number of Agents	12	12	12	12
	Discovery Time (s)	–	–	–	–
Search Cycle (9s)	Number of Agents	5.9	12	12	6
	Discovery Time (s)	11.7	–	–	11.5
Search Cycle (6s)	Number of Agents	6	12	12	6
	Discovery Time (s)	11.7	–	–	11.6

Table VIII
DETECTION TIME ON FLOOR MAP 2 BY WALKING SPEED 1.5M/S

Walking Speed(1.5m/s)		Ripple	Stationary Net (n=2)	Stationary Net (n=3)	Stationary Net(n=4)
Search Cycle (12s)	Number of Agents	14	14	10	12.2
	Discovery Time (s)	49.5	–	31.8	32.6
Search Cycle (9s)	Number of Agents	13.8	10	10	13
	Discovery Time (s)	33.7	32.3	32.3	33
Search Cycle (6s)	Number of Agents	13.9	10	13.2	14
	Discovery Time (s)	32.2	32	32.4	33

Table IX
DETECTION TIME ON FLOOR MAP 2 BY WALKING SPEED 2M/S

Walking Speed(2m/s)		Ripple	Stationary Net (n=2)	Stationary Net (n=3)	Stationary Net(n=4)
Search Cycle (12s)	Number of Agents	14	14	10	10
	Discovery Time (s)	–	–	31.1	25.3
Search Cycle (9s)	Number of Agents	14	14	10	13
	Discovery Time (s)	35.9	–	25.6	25.2
Search Cycle (6s)	Number of Agents	13.8	10	13	14
	Discovery Time (s)	24.8	25.3	24.8	25.8

Table X
DETECTION TIME ON FLOOR MAP 2 BY WALKING SPEED 3M/S

Walking Speed(3m/s)		Ripple	Stationary Net (n=2)	Stationary Net (n=3)	Stationary Net(n=4)
Search Cycle (12s)	Number of Agents	14	14	14	9.8
	Discovery Time (s)	–	–	–	18
Search Cycle (9s)	Number of Agents	14	14	14	10
	Discovery Time (s)	–	–	–	18.2
Search Cycle (6s)	Number of Agents	14	14	10.4	12.4
	Discovery Time (s)	25.9	–	18.2	19

VI. Conclusion

We propose the Ripple detection method and the Stationary net detection method. These are examined using a image processing simulator, because an individual can not be accurately identified in the current image processing. And in addition, the image processing simulator can be stabilized the accuracy of image processing and the simulator can be effective to examine each method correctly. These detection methods can search a lost target but a search cycle has to be within $(\textit{walking speed} \times \textit{distance between cameras})$. These methods can be efficient to detect a target if the search cycle is near the walking speed. A mobile agent can keep tracking a target by using these detection methods if the agent lose the target. In addition, from the examination results, the Stationary net detection method can detect a target faster than the Ripple detection method. And the Stationary net detection method can use smaller number of agents than the Ripple detection method. Because the Ripple detection method searches a target by widening a search gradually but the Stationary net detection method can widen a search efficiently by the Neighbor node determination algorithm.

We will research more efficient detection to improve the automatic human tracking system. And we are considering to compute a movement direction of a captured target, to improve efficiency of tracking by using the direction information to the tracking, and to improve the detection by using the number of route which is element of neighbor node information E. In addition, the accuracy of image processing has to be improved more to track a target more accurately. We are considering to improve image processing program by using PCA-SIFT [23] or SURF [24] algorithm.

Acknowledgment

The authors would like to thank Tadaaki Shimizu, Yusuke Hamada, Naoki Ishibashi, Shinya Iwasaki, Hirotoshi Okumura, Masato Hamamura, Shingo Iiyama in Tottori university.

References

[1] K. Terashita, N. Ukita, and M. Kidode, "Efficiency improvement of probabilistic-topological calibration of widely distributed active cameras," *IPSJ SIG Technical Report*, vol. 2009–CVIM–166, pp. 241–248, 2009.

[2] Y. Kawaguchi, A. Shimada, D. Arita, and R. Taniguchi, "Object trajectory acquisition with an active camera for wide area scene surveillance," *IPSJ SIG Technical Report*, vol. 2008–CVIM–163, pp. 1306–1311, 2008.

[3] H. Yin and I. Hussain, "Independent component analysis and nongaussianity for blind image deconvolution and deblurring," *Integrated Computer-Aided Engineering*, vol. 15, no. 3, pp. 219–228, 2008.

[4] U. M. Erdem and S. Sclaroff, "Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements," *CVIO2006*, vol. 103, no. 3, pp. 156–169, 2006.

[5] Y. Yao, C. H. Chen, B. Abidi, D. Page, A. Koschan, and M. Abidi, "Sensor planning for automated and persistent object tracking with multiple cameras," *CVPR2008*, 2008.

[6] ——, "Sensor planning for ptz cameras using the probability of camera overload," *ICPR2008*, 2008.

[7] A. Nakazawa, S. Hiura, H. Kato, and S. Inokuchi, "Tracking multiple persons using distributed vision systems," *Journal of Information Processing Society of Japan*, vol. 42, no. 11, pp. 2699–2710, November 2001.

[8] T. Darrell, G. Gordon, M. Harville, and J. Woodfill, "Integrated person tracking using stereo, color, and pattern detection," *International Journal of Computer Vision*, vol. 37, no. 2, pp. 175–185, June 2000.

[9] D. B. Lange and M. Oshima, "Seven good reasons for mobile agents," *Communications of the ACM*, vol. 42, no. 3, pp. 88–89, 1999.

[10] R. S. Gray, G. Cybenko, D. Kotz, R. A. Peterson, and D. Rus, "D' agents: Applications and performance of a mobile-agent system," *Software: Practice and Experience*, vol. 32, no. 6, pp. 543–573, 2002.

[11] S. Motomura, T. Kawamura, and K. Sugahara, "Maglog: A mobile agent framework for distributed models," in *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, 2005, pp. 414–420.

[12] T. Kawamura, S. Motomura, and K. Sugahara, "Implementation of a logic-based multi agent framework on java environment," in *Proceedings of IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005, pp. 486–491.

[13] G. Cabri, L. Leonardi, and F. Zambonelli, "Mobile-agent coordination models for internet applications," *Computer*, vol. 33, no. 2, pp. 82–89, February 2000.

[14] G. Valetto, G. Kaiser, and G. S. Kc, "A mobile agent approach to process-based dynamic adaptation of complex software systems," *Lecture Notes in Computer Science*, vol. 2077, pp. 102–116, 2001.

[15] N. R. Jennings, "An agent-based approach for building complex software systems," *Communications of the ACM*, vol. 44, no. 4, pp. 35–41, April 2001.

[16] D. Monticolo, V. Hilaire, S. Gomes, and A. Koukam, "A multi-agent system for building project memories to facilitate the design process," *Integrated Computer-Aided Engineering*, vol. 15, no. 1, pp. 3–20, January 2008.

[17] H. Kakiuchi, Y. Hamada, T. Kawamura, T. Shimizu, and K. Sugahara, "To realize automatic human tracking system based on mobile agent technologies," in *Proceedings of the 59th Chugoku branch union convention of the Institute of Electrical Engineers of Japan and Information Processing Society of Japan*. Institute of Electrical Engineers of Japan and Information Processing Society of Japan, 2008, p. 485.

[18] Y. Hamada, S. Iwasaki, H. Kakiuchi, T. Kawamura, and K. Sugahara, "Pursuit methods for automatic human tracking system based on mobile agent technologies," in *Proceedings of the 59th Chugoku branch union convention of the Institute of Electrical Engineers of Japan and Information Processing Society of Japan*. Institute of Electrical Engineers of Japan and Information Processing Society of Japan, 2008, p. 486.

[19] N. Ishibashi, Y. Hamada, H. Kakiuchi, T. Shimizu, T. Kawamura, and K. Sugahara, "Feature extraction method for automatic human tracking system based on mobile agent technologies," in *Proceedings of the 59th Chugoku branch union convention of the Institute of Electrical Engineers of Japan and Information Processing Society of Japan*. Institute of Electrical Engineers of Japan and Information Processing Society of Japan, 2008, p. 418.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[21] H. Kakiuchi, T. Kawamura, T. Shimizu, and K. Sugahara, "An algorithm to determine neighbor nodes for automatic human tracking system," in *IEEE International Conference on Electro/Information Technology*. IEEE, 2009, pp. 96–102.

[22] Open Service Gateway Initiative Alliance, *OSGi Alliance Specifications OSGi Service Platform Release 1*, last access May 2011. [Online]. Available: http://www.osgi.org/Specifications/HomePage

[23] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," *Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 506–513, 2004.

[24] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

PIGA-HIPS: Protection of a Shared HPC Cluster

M. Blanc*, J. Briffaut, D. Gros, C. Toinard
Laboratoire d'Informatique Fondamentale d'Orléans
*CEA, DAM, DIF F-91297 Arpajon, France, mathieu.blanc@cea.fr
ENSI de Bourges – LIFO, 88 bd Lahitolle, 18020 Bourges cedex, France
{jeremy.briffaut,damien.gros,christian.toinard}@ensi-bourges.fr

***Abstract*—Protecting a shared High Performance Computing cluster is still an open research problem. Existing solutions deal with sand-boxing and Discretionary Access Control for controlling remote connections. Guaranteeing security properties for a shared cluster is complex since users demand an environment at the same time efficient and preventing confidentiality and integrity violations. This paper proposes two different approaches for protecting remote interactive accesses against malicious operations. Those two approaches leverage the SELinux protection. They have been successfully implemented using standard MAC from SELinux, and guarantee supplementary security properties thanks to our PIGA HIPS. The paper compares those two different approaches. It presents a real use case for the security of a shared cluster that allows interactive connections for users while preventing confidentiality and integrity violations. That paper takes advantage of previous works and goes one step further for protecting shared clusters against malicious activities. It proposes a new framework to share a cluster among partners while guaranteeing advanced security properties. This solution aims to prevent complex or indirect malicious activities that use combinations of processes and covert channels in their attempt to bypass the required properties.**

Keywords - Security Property, Mandatory Access Control, High Performance Computing Security.

I. Introduction

Protecting a HPC cluster [1] against real world cyber threats is a critical task nowadays, with the increasing trend to open and share computing resources. As partners can upload data that is confidential regarding other partners, a company managing a shared cluster has to enforce strong security measures. It has to prevent both accidental data leakage and voluntary data stealing.

Security of the clusters accesses usually relies on Discretionary Access Control (DAC) and sand-boxing such as chroot, BSD Jail or Vserver. The DAC model has proven to be fragile [2]. Moreover, virtual containers do not protects against privilege escalation where users can get administration privileges in order to compromise data confidentiality and integrity.

Mandatory Access Control (MAC) can be used to confine the various cluster populations. MAC such as in NSA's Security-Enhanced Linux (SELinux) provides a powerful protection scheme. However, defining efficient SELinux policies is complex. Moreover, advanced security properties cannot be enforced by that approach. For example, SELinux in its current design does not control information flows involving multiple processes and ressources.

Solutions such as [3] propose enforcement of advanced security properties for SELinux. However, efficient protection of remote accesses has not yet been proposed using those kinds of solutions. Moreover, specific contexts of use such as a cluster shared between various entities require a new protection scheme since the protection must scale well.

That paper answers those two questions. First, it presents two different solutions for the protection of a shared cluster against malicious usage. Second, it discusses the advantages and presents the solution chosen to protect a large scale cluster such as the ones deployed by the CEA. Finally, it shows how to compute the remaining risks associated with a given SELinux policy, how to analyze the remaining risks and how to prevent them. The result is that, however complete, the proposed SELinux policy still cannot prevent about a million of indirect illegal activities. But, the PIGA HIPS enables to prevent against these remaining risks. Then, the paper presents a real case study showing realistic scenarii of attacks, and how the PIGA HIPS can prevent them.

II. Related Work

High performance computing architectures are extremely specialized, compared to general computing facility. As such, they present specific security issues and properties. As outlined by William Yurcik [4], these issues must be addressed in a way that is relevant, with a combination of general techniques and one that are specific to cluster architectures.

Sandboxing such as [5] or [6] provides a mean to confine processes. However, in the context of a shared cluster where processes can communicate and share files, the confinement cannot be hardened and information can flow between processes and resources.

Under Linux, there are at least four security models available to ensure a Mandatory Access Control policy: SELinux, grsecurity, SMACK and RSBAC. But none of these solutions can ensure a large set of security properties. In the best case, they can ensure one or two limited properties such as the Bell and LaPadula confidentiality or the Biba integrity. Under the BSD family, solutions such as Trusted BSD (available within the following Operating Systems: FreeBSD, OpenBSD, MacOSX, NetBSD) provide more or less the same kind of a Mandatory Access Control as SELinux. But, again, they fail to ensure the large majority of requested security properties.

The major limitation is related to indirect information flows, allowed by the considered protection policies, that enable to violate a security property. All those approaches fail to correctly manage indirect information flows, consequently authorizing illegal activities.

Several studies address how to manage indirect information flows within an Operating System. The HiStar Operating System [7] associates each object or subject with an information flow level. The problem of HiStar is that it is very close to the the Biba integrity model and suffers from the same limitations. The Flume Operating System [8] is very close to HiStar. However, Flume does not control efficiently the information flows.

Asbestos [9] reuses the idea of HiStar by considering four different levels of information. The protection rules can only express pairwise relationship patterns. Again, information flows involving multiple interactions and processes cannot be controlled easily.

Works about the enforcement of dynamic policies such as [10], [11], generally consider how to detect simple conflicts within dynamic policies. For example, they detect if it is safe to remove or add a role or a context, otherwise the considered access control could become invalid, conflicting or not supported. So, they address conflicting rules but do not enforce a large set of security properties.

Briffaut and al. [3] presents how to reinforce the security of SELinux MAC policies. However, security properties for protecting a HPC cluster must be defined. Moreover, the protection system must scale well in order to minimize the performance overhead and ease the deployment.

III. Security objectives

In this section, we present our security goals regarding our experimental shared HPC clusters. These goals are the basis of our security policy, and thus of our SELinux configuration. They can be resumed in five points:

- Ensure the confidentiality of data uploaded by partners;
- Confine user profiles and services so that a malicious elevation of privileges does not compromise the security of the operating system;
- Differentiate public SSH access and administrator SSH access;

The following subsections give details on each point.

A. User containers and data confidentiality

Users from the same projects should be able to exchange files freely. Hence there is a particular set of Unix groups called "containers". These containers represent people working on the same project or users that are granted access by the same administrative procedure (for example a national research agreement). In these containers, accidental leakage of information due to incorrect permissions is considered harmless.

Definition 3.1 (Container): In a container A, with $(a_1, a_2) \subset A$, a_1 accessing a file f belonging to a_2 does not break the confidentiality of file f.

which means, in terms of confidentiality:

Definition 3.2 (Confidentiality): Data confidentiality means that a user a from container A must not be able to read a file f belonging to a user b from container B, whatever permissions are set on f.

Of course, this does not mean that any user can access all the files of all other users in the same container. Typically, aMAC mechanism will confine users in their container, and then inside a container users can restrict access to their own files with DAC permissions.

B. Confined users and services

Users and services should be confined in order to prevent any tampering with the security mechanisms. A first example is a malicious hacker exploiting a flaw in a network service in order to gain administrative access to a login node. Exploiting a flaw should not allow him to break the data confidentiality of another container. Another example is a legitimate user downloading a malicious code from the Internet and using it to gain administrative privileges on his node. Even if this succeeds, this user should not be able to access files outside his container.

Definition 3.3 (Confinement): Any person gaining administrative privileges on a system must not be able to break the confidentiality property, either legitimate user or external attacker.

C. SSH access

There should be two different points of access on the cluster nodes: a public access for standard users, and an administrative access reserved to system administrators. These accesses are always setup with a ciphered protocol like SSH. Even in the case a vulnerability is exploited in the server and gives an administrative access to an attacker, the public access should never allow users to configure the security mechanisms. Only the administrative access should.

Of course interactive user access is not always enabled. For example, there are some parts of the clusters like computing nodes where standard user access should be disabled. These nodes should only be accessed through the batch scheduler. The same restriction goes for the storage nodes, accessible only through mounted network file systems, and so on. These are only examples, each cluster has its specific areas.

IV. SELinux Solutions of Protection

A. Solution 1: SSH users confinement with chroot and SELinux

When the SSH daemon receives a connection the user is authenticated, it forks and executes the user's shell from `/etc/passwd`. Our confinement system provides a chroot confinement for this shell, strengthened by SELinux rules via a SELinux module. First, we create a Linux sub tree in which the SSH daemon will chroot the user. The main idea is to build one confinement tree per user, and each user has different SELinux types. We use the base types defined for / tree, adding the username of the user linked to the confinement. For example, if we confine Bob in `/cage`, files in `/cage/etc` will have

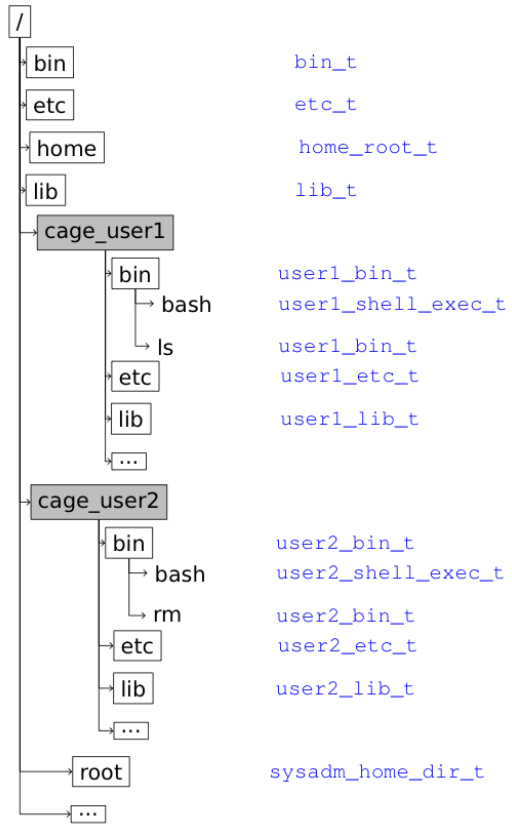


Fig. 1. Jail and SELinux contexts.

the `Bob_etc_t` type, as files in `/etc` have the `etc_t` type. See figure 1 for an example of SELinux types we use.

The goal of our SELinux module is to make an automatic transition of the user confined shell to a unique type, and to give rights to this type to interact only with objects that own a type in the user's sub tree. So, each user's processes and objects have a different SELinux type and consistent protection rules prevent from unauthorized accesses. Even in case of chroot or application corruption, a process (e.g. user1 subject) is prevented from accessing unauthorized resources (e.g. user2 objects).

Here are the main steps to confine a user. The first step is to create the sub tree. Jail is a simple Linux tool (not to be confused with FreeBSD Jail) to build such a sub tree. Then, we build the SELinux module to strengthen the confinement. We write the source code of the module, compile it and load it to SELinux. The next step is to configure OpenSSH to chroot this user in the requested sub tree, and apply the correct SELinux labels to the user file tree (relabeling operation). As the SELinux module has been loaded before relabeling the tree, it will get types we defined in our policy module, not default types.

A bash script provides automation for these different steps. The script just needs the user name and the path you want to confine him in. For our tests we used the SSHJail patch

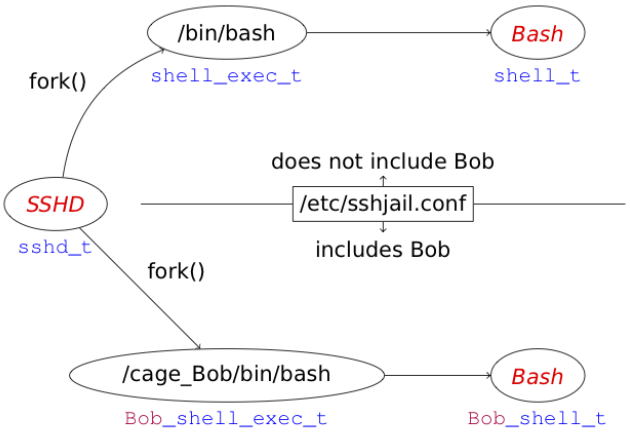


Fig. 2. Jail and SELinux transitions.

for OpenSSH, which use the `/etc/sshjail.conf` file to define user's chroot. The script must be adapted if using another system. Of course, the script checks if this user is already confined in another sub tree or if the sub tree is already used to confine another user. The SELinux rules defined for the modules are very restrictive, they just allow the user to login and run a few commands such as `ls`. Other commands can be added on demand based on application requirements.

This system requires to set up each user on the machine. So, our script needs to be executed after the `useradd` command in order to confine every user. It is a good thing to gather these two steps into a single one. The solution is to provide an alias for the `useradd` command that runs the two steps. A SELinux rule prevents the real `useradd` from being run directly. Thus, only the alias is allowed for execution.

B. Solution 2: SSH users confinement with Port differentiation and SELinux

The idea is to have different SSH ports for each user categories. In the sequel, we consider only two types of users. One has the context `ccc_guest_t` and offers a restricted access. The other one is `unconfined_t` which gives full privilege access. To separate these two kinds of interactive accesses, we introduce two new contexts for two SSH servers, `sshd_public_t` and `sshd_admin_t`. The first one offers restricted access whereas the second one gives a privileged access.

This is implemented in two different SELinux policy modules described in the following parts: `ccc_guest` and `sshd_admin`.

1) `ccc_guest`: By default, all the users are placed in the context `(user_u, system_r, unconfined_t)`. The goal of our SELinux module is to provide two confined user profiles: `ccc_guest` and `ccc_xguest`. The first one is associated to SSH connections, and the second one is associated to X11 sessions. They are originally derived from the `guest` and `xguest` profiles of Fedora 10, provided by Dan

Walsh [12]. They were subsequently adapted to our specific needs.

```
1  unprivileged_user(ccc_guest_t, ccc_guest);
```

This template deals with the creation of the basic rules for the ccc_guest_t type. When the user logs on the system, he receives a set of access rights that allows him to perform basic actions. This unprivileged_user template also enables to use a part of the network (important for the use of SSH).

```
1  userdom_restricted_xwindows_user_template(ccc_xguest);
2  use_kde(ccc_xguest_t);
```

The xguest profile derived of Fedora 10 allows us to define rules for the X11 forwarding in SSH. There are specific rules for the use of X11 and graphical environment such as KDE.

```
1  corecmd_shell_domtrans(sshd_public_t, ccc_guest_t);
```

This template enables the ccc_guest_t type to interact with the sshd_public_t. The corecmd_shell_domtrans allows the public SSH context to transit to the guest type, and only this one. That way, the public SSH server only gives restricted access.

```
1  use_local_home_dir(ccc_guest);
```

The use_local_home_dir template allows the user to manipulate and more especially to create, manage file permissions and relabel certain types of files and directories, for example the user_home_t type.

For the use of NFS and LUSTRE file system, specific rules have been defined.

These different templates allow us to use SSH with a specific type (sshd_public_t) and to confine the shell of the user. The objective of these confined user profiles is to limit the administrative privileges accorded to users to the minimum. For example, a standard user logged on the system via SSH will have the context (ccc_guest_u, ccc_guest_r, ccc_guest_t). Trying to exploit a vulnerability in any system command or service, he may obtain a root access, but he will still have the same confined SELinux context and will not be able to take advantage of this root access.

Moreover, several rules protect the system against malicious code execution by the user. For example, the stack is protected against the execution with the following rule.

```
1  allow $1_t self:process ~{ setcurrent setexec execmem execstack };
```

2) sshd_admin: By default, the SSH server is not associated to a particular context, it is actually executed in the unconfined_t type. As the SSH service must be accessible to all the cluster partners from the Internet, it is heavily exposed to attacks. This policy module answers two goals. First, we want to confine the SSH server so that if an attacker exploits a vulnerability in it, he will only reach a confined profile. Secondly, the attainable SELinux roles from the SSH server should be limited to what is strictly necessary. That is why we create two different contexts for two SSH servers running at the same time (on different TCP ports): sshd_public_t and sshd_admin_t.

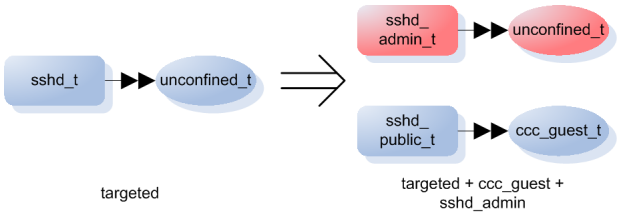


Fig. 3. Two levels of SSH access.

The first context is for the public SSH access. We call it "public" but it may already be filtered at the network level. This context can only transit to the ccc_guest_t type. The second context is reserved for administrative access, and can transit to the unconfined_t type. This is illustrated in figure 3.

The profile for the admin is close to the ccc_guest profile. It can perform several operations on files and directories. The policy adds the right for the admin to list and read the root directory. The admin can also use X11.

```
1  unprivileged_user(ccc_admin_t, ccc_admin);
2  userdom_restricted_xwindows_user_template(ccc_admin);
```

The admin is able to transit to the unconfined_t thanks to this specific rule whereas there is a rule that denies the transition for the user to transit to the unconfined_t.

```
1  neverallow sshd_public_t unconfined_t:process { transition };
2  allow sshd_admin_t unconfined_t:process { transition };
```

The admin has to be able to administrate the ccc_guest. A special program named xbe is used. The use_xbe template provides the set of rules allowing to manage the guest processes and change the permissions on the guest files.

```
1  use_xbe(ccc_admin, ccc_guest);
```

For example, the following rule is included in the use_xbe template. This rule enables the ccc_admin_t subject to control the ccc_guest_t process.

```
1  allow ccc_admin_t ccc_guest_t:process { siginh rlimitinh noatsecure };
```

C. Discussion of the two SELinux solutions

When a user logs on the cluster, he receives his allowed set of permissions. He will be able to access only the files that are allowed to him.

Our intended protection is to provide containers. For the first solution, containers are associated with the user identities. Each user can be seen as a container. In the context of a shared cluster, it is not a scalable approach since the users cannot easily share data. A complex SELinux policy must be defined in order to allow the required sharing. However, that approach enables a complete control of the resources's accesses. For the second solution, containers are associated with the server port. In the sequel, we consider only two SSH server ports. But, in the context of multiple partners sharing the cluster, each partner would use a dedicated server. That approach scales

better since a container is associated to each partner. So, one does not need to define complex sharing policies.

Our main objective is to prevent against indirect information flows i.e. covert channels. While offering a very robust protection in case of an application security vulnerability, SELinux cannot control those flows. In the following section, we show how the PIGA security tools are used to assess those indirect flows. Dedicated security properties are proposed to control the flows of our second solution.

V. Analysis of the Cluster Protection

In order to analyze and to reinforce the security provided by the SELinux policy of our second solution, several security properties are proposed using the Security Property Language defined in [3]. The advantages of our SPL are 1) to enumerate the remaining risks within a given SELinux policy and 2) to provide a mean to prevent these risks. That section will describe the first point in order to analyse the remaining risks in our SELinux policy. The second point will be developed latter on in the paper when considering the enforcement of the Security Properties in order to prevent againts these remaining risks.

A. Security Properties

Using the SPL described in [13], several security properties templates are proposed. In order to analyse the remaining risks within our SELinux MAC policy, that section defines dedicated security templates such as *transitionsequence*, *cantransit*, *dutiesseparationbash* and *dutiesseparationreadwrite*. That section shows how the usage of these security templates enables to analyze the security risks of our second solution. Each security template can be considered as a generic security objective. A template enables to define an instance of the considered security property associated with relevant security contexts. Each instance of a security property corresponds to a security objective that the target operating system must satisfies.

1) Templates of Security Properties: The first template of security property enables to protect against confidentiality violation between a source security context $sc1$ and an object security context $sc2$. Thus, one can prevent both direct and indirect information flows from $sc2$ to $sc1$.

```
1  define confidentiality( $sc1 IN SCS, $sc2 IN SCO ) [
2          ST { $sc2 > $sc1 },
3                  { not(exist()) };
4          ST { $sc2 >> $sc1 },
5                  { not(exist()) };
6  ];
```

The second template of security property enables to protect against integrity violation from a source security context $sc1$ against an object security context $sc2$. Thus, one can prevent both direct and indirect integrity violations from $sc1$ against $sc2$.

```
1  define integrity( $sc1 IN CSS, $sc2 IN CSO ) [
2          Foreach $eo IN is_write_like(IS)
3                  ST { $sc1−> { $eo } $sc2 } ,
4                          { not(exist()) };
5          Foreach $eo IN is_write_like(IS)
6                  ST { $sc1 => { $eo } $sc2 } ,
7                          { not(exist()) };
8  ];
```

The third template of security property enables to detect all the transitions to the existing source security contexts. That template is usually not used to prevent transitions but to detect the transitions carried out on the target system.

```
1  define transitionsequence( ) [
2          Foreach $sc1 IN { system_u:system_r:init_t }, Foreach $sc2
                  IN CSS
3           ST { $sc1 __> $sc2 },
4                  { not(exist()) };
5  ];
```

The fourth template of security property enables to prevent transitions from a source security context $SCFROM$ to another source security context $SCTO$. Thus, one can prevent processes to transit into specific contexts to get illegal privileges.

```
1  define cantransit( $SCFROM IN CS, $SCTO IN CS) [
2          Foreach $sc1 IN $SCFROM, Foreach $sc2 IN $SCTO
3           ST { $sc1 __> $sc2 },
4                  { not(exist()) };
5  ];
```

The fifth template of security property enables to prevent indirect activities from violating the existing SELinux direct policy. Thus, a process cannot get indirect privileges that are conflicting with the allowed direct SELinux permissions.

```
1  define consistentaccess($sc1 IN CS, $sc2 IN CS) [
2          ST { $sc1 >>> $sc2 },
3                          { exist[$sc2 > $sc1] };
4  ];
```

The sixth template of security property enables to prevent bash activities to write and then read scripts. Thus, one can prevent attacks consisting in using bash to execute illegal scripts.

```
1  define dutiesseparationbash( $sc1 IN CS ) [
2          Foreach $eo1 IN is_write_like(IS), Foreach $eo2 IN
                  is_execute_like(IS), Foreach $eo3 IN is_read_like(IS),
3          Foreach $sc2 IN $CSONE, Foreach $sc3 IN CS,
4          Foreach $a1 IN ACT, Foreach $a2 IN ACT
5                  ST { ( [ $a2 := $sc1 −> { $eo3 } $sc2 ] o ( [ $a1 :=
                          $sc1 −> { $eo2 } $sc3 ] o $sc1 −> { $eo1 }
                          $sc2) ) } ,
6                          { INHERIT($a2 , $a1) };
7  ];
```

The seventh template of security property enables to prevent malicious activities to write and then read files. Thus, one can prevent attacks consisting in writing data in order to forward the produced information.

```
1  define dutiesseparationreadwrite( $sc1 IN CS ) [
2          Foreach $eo1 IN is_write_like(IS), Foreach $eo2 IN
                  is_read_like(IS), Foreach $sc2 IN $CSRW
3                  ST { ($sc1 −> { $eo2 } $sc2 o $sc1 −> { $eo1 }
                          $sc2) } ,
4                          { not( exist() ) };
5  ];
```

2) Analysis of SElinux through instances of the security properties: The templates defined previously enable to define several instances of the security properties for protecting the administrator role, as in the following listing. The first property prevents the admin role to transit to the guest role. The second, third and fourth properties force the admin role to satisfy various separation of duties. For example, the admin role cannot uses bash to execute illegal scripts. He can only execute scripts legally installed on the target SELinux. The fifth

and sixth properties prevent against integrity violation carried out from the admin role. The seventh property prevents the admin role from indirectly getting permissions that are not available in the SELinux policy. The eighth property prevents the admin role to violate the confidentiality of the guest role either directly or indirectly.

Listing 1. Security properties for admin

```
1 cantransit( $SCFROM:=".*:ccc_admin_r:.*", $SCTO:=".*:ccc_guest_r
      :.*" );
2 dutiesseparation( $sc1:=".*:ccc_admin_r:.*" );
3 dutiesseparationbash( $sc1:=".*:ccc_admin_r:.*" );
4 dutiesseparationreadwrite( $sc1:=".*:ccc_admin_r:.*" );
5 integrity( $sc1:=".*:ccc_admin_r:.*", $sc2:=".*:.*:.*_exec_t" );
6 integrity( $sc1:=".*:ccc_admin_r:.*", $sc2:=".*:.*:.*etc_t" );
7 consistentaccess($sc1:=".*:ccc_admin_r:.*" );
8 confidentiality( $sc1:=".*:ccc_admin_r:.*", $sc2:=".*:ccc_guest_r:.*" );
```

Several similar properties are defined for protecting the guest role. Those properties enable to prevent the guest role to violate the confidentiality of the admin role.

Listing 2. Security properties for guest

```
1 cantransit( $SCFROM:=".*:ccc_guest_r:.*" , $SCTO:=".*:ccc_admin_r
      :.*");
2 dutiesseparation( $sc1:=".*:ccc_guest_r:.*" );
3 dutiesseparationbash( $sc1:=".*:ccc_guest_r:.*" );
4 dutiesseparationreadwrite( $sc1:=".*:ccc_guest_r:.*" );
5 integrity( $sc1:=".*:ccc_guest_r:.*", $sc2:=".*:.*:.*_exec_t" );
6 integrity( $sc1:=".*:ccc_guest_r:.*", $sc2:=".*:.*:.*etc_t" );
7 consistentaccess($sc1:=".*:ccc_guest_r:.*" );
8 confidentiality( $sc1:=".*:ccc_guest_r:.*", $sc2:=".*:ccc_admin_r:.*" );
```

B. SELinux Policy Analysis with PIGA

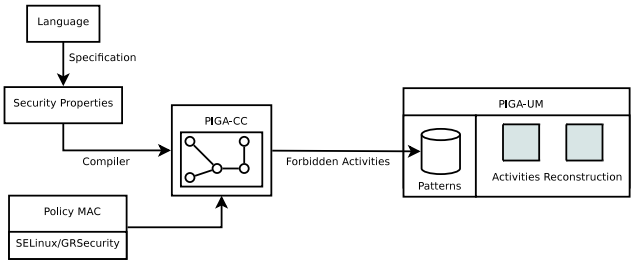


Fig. 4. Process of analyzing a SELinux policy and generating the patterns violating the security properties

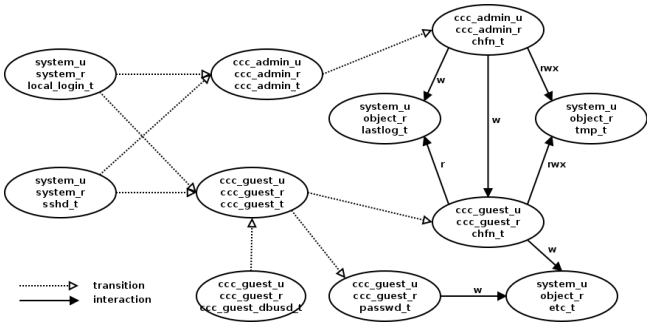


Fig. 5. Interaction graph

The objectives are first to enumerate the possible risks included into a given SELinux policy and second to provide a mean to prevent against these remaining risks. For these purposes, Figure 4 shows how to analyze a SELinux policy in order to enumerate the remaining risks. A compiler, PIGA-CC, a module of PIGA, is used to enumerate from a given MAC policy (i.e. a SELinux policy) the set of the illegal activities associated with requested security properties. This section only addresses the first point i.e. how the remaining risks can be enumerated. Latter on in the paper, we will address the way the enumerated remaining risks enable to guarantee the security properties.

The PIGA-CC module builds a graph that represents the access control policy (the SELinux policy). For each security property instanciation, PIGA-CC enumerates paths into this graph. Each enumeration, a combination of paths, corresponds to a possible violation of a required security property starting from the considered SELinux policy. Thus, the compiler enumerates all the forbidden activities i.e. all the sequences of system calls allowing the violation of a considered security property. The compiler sends those illegal activities to PIGA-UM in order to prevent the violation of the requested security properties.

Figure 5 gives an example of the sub-graph computed by PIGA-CC for the SELinux policy considered in this paper. In this graph, each edge between two contexts sc_1 and sc_2 corresponds to a direct interaction $sc_1 \rightarrow sc_2$, and each path between two contexts sc_1 and sc_n corresponds to an indirect interaction $sc_1 \Rightarrow sc_n$. This graph enables to enumerate all the terminals of the SPL language.

For example, when PIGA-CC analyses the confidentiality property:

```
1 define confidentiality( $sc1 IN SCS, $sc2 IN SCO ) [
2         ST { $sc2 > $sc1 },
3                 { not(exist()) };
4         ST { $sc2 >> $sc1 },
5                 { not(exist()) };
6 ];
```

It extracts the edge between sc_1 and sc_2 and then it generates a violation for each operation between these two contexts that correspond to a possible information transfer, i.e., a read operation between sc_1 and sc_2 or a write operation between sc_2 and sc_1. Next, PIGA-CC enumerates the set of paths between sc_1 and sc_2 where each edge is an information transfer correctly oriented.

Let us give an example for the Figure 5 and the following security property:

```
1 confidentiality( $sc1:=".*: ccc_guest_r:.*", $sc2:=".*:ccc_admin_r:.*" );
```

As shown in Figure 5, an information transfer is possible between the context $ccc\_admin\_u : ccc\_admin\_r : ccc\_admin\_t$ and $ccc\_guest\_u : ccc\_guest\_r : ccc\_guest\_t$ using the intermediate object $system\_u : object\_r : lastlog\_t$. Thus, this policy does not respect this confidentiality property since there is an activity allowing a forbidden information transfer between $ccc\_admin\_u : ccc\_admin\_r : ccc\_admin\_t$ and $ccc\_guest\_u : ccc\_guest\_r : ccc\_guest\_t$. As presented in the sequel all the forbidden activities can be used to prevent the considered flows i.e. to guarantee the requested properties.

C. Results of the SELinux policy analysis

This section presents the results provided using PIGA-CC. These results corresponds to the security analysis of the SELinux policies available for our second solution. These results are presented in two tables. The table I describes the analysis for the administrator role and the table II for the guest role. These tables are divided into two columns.

The first one deals with direct activities.These activities are blocked by SELinux. The second one deals with indirect activities. These flows cannot be prevented by SELinux but they can be blocked byPIGA-SP an extended MAC approach reusing the enumerated activities. The PIGA-SP MAC approach will be presented in the sequel.

For example, in the table I, PIGA-CC was able to find 4 direct activities dealing with integrity property and 8 indirect activities.

Table I shows, for the admin role, all the direct illegal activities and all the indirect illegal activities. This table is based on the listing 1. The direct illegal activities can be prevented simply by a modification of the SELinux policy. But, PIGA-SP MAC can prevent all those direct and indirect violations. As shown in table I, the confidentiality property can prevents against 548.858 potentially illegal activities. Those illegal activities are vectors that enable intruders to develop exploits against the SELinux protection by using for example a combination of buffer overflows and covert channels.

TABLE I
RESULT FOR THE ADMINISTRATOR ROLE

Security property	direct activities	indirect activities
cantransit	0	0
dutiesseparation	0	469
dutiesseparationbash	0	124288
dutiesseparationreadwrite	0	1191
integrity	0	0
integrity	4	8
consistentaccess	0	1474
confidentiality	98	548858

Table II shows the illegal activities for the guest role.This table is based on the listing 2. It shows that PIGA-SP MAC can prevent against about 1 million of illegal activities that could compromise the SELinux protection. Those results demonstrate that SELinux is not able to guarantee advanced security properties such as the ones required for protecting a shared cluster. In contrast with SELinux, PIGA-SP MAC can efficiently enforce all the requested properties. The sequel presents the way PIGA-SP reuses the illegal activities for preventing the violation of the requested properties. PIGA-SP is an advanced Host Intrusion Prevention System. But the illegal activities can be used to provide an Host Intrusion Detection System. Before explaining the internals of the PIGA-SP MAC approach, let us first give an example of typical scenarii of attacks allowed by the SELinux policy.

TABLE II
RESULT FOR THE GUEST ROLE

Security property	direct activities	indirect activities
cantransit	0	0
dutiesseparation	0	1118
dutiesseparationbash	0	328362
dutiesseparationreadwrite	0	2530
integrity	0	0
integrity	8	16
consistentaccess	0	3026
confidentiality	96	728214

D. Case study of remaining risks within the SELinux policy

Amoung the millions of potentially illegal activities, let us choose some of them to show the typical scenariis of attacks that PIGA can prevent.

Regarding the confidentiality of the admin role, the line 2 of the following listing shows a direct activity where the `chfn` application (that changes your finger information) uses a covert channel, i.e. writing in a fifo file, with the guest role. One can imagine an exploit against the `chfn` application using that direct covert channel to make the admin's data flow to the guest role. The line 3 shows an indirect activity, where the admin role writes a log file that is then read by the guest role. If such an activity occurs, PIGA-SP guarantees that the reading of the log file will fail.

```
1  confidentiality( $sc2:=".*:ccc_admin_r:.*", $sc1:=".*:ccc_guest_r:.*" );
2  2$55 : ccc_admin_u:ccc_admin_r:chfn_t −( fifo_file { write } )−>
       ccc_guest_u:ccc_guest_r:chfn_t
3  3$280219 : ccc_admin_u:ccc_admin_r:ccc_admin_t −( file { append
       write } )−> system_u:object_r:lastlog_t ; ccc_guest_u:
       ccc_guest_r:ccc_guest_t −( file { read } )−> system_u:object_r
       :lastlog_t
```

Regarding the confidentiality of the guest role, the line 2 of the following listing shows a direct activity where the SELinux policy enables the guest role to use a covert channel, i.e. writing in a fifo file, with the admin role. The line 3 shows an indirect activity, where the guest role writes a temporary file that is then read by the admin role. It is an indirect covert channel that PIGA can prevent. Thus, the reading of the temporary file by the admin role will fail.

```
1  confidentiality( $sc1:=".*:ccc_admin_r:.*", $sc2:=".*:ccc_guest_r:.*" );
2  0$90 : ccc_guest_u:ccc_guest_r:ccc_guest_t −( fifo_file { write } )−>
       ccc_admin_u:ccc_admin_r:ccc_admin_t
3  1$471971 : ccc_guest_u:ccc_guest_r:ccc_guest_t −( file { append
       write } sock_file { append write } )−> system_u:object_r:tmp_t ;
        ccc_admin_u:ccc_guest_r:ccc_guest_t −( file { read } sock_file
        { read } )−> system_u:object_r:tmp_t ;
```

Regarding the integrity of the admin role, the line 2 of the following listing shows a direct activity where the `chfn` application can write the `/etc` files. One can imagine to use `chfn` to get an admin role and thus modifies the configuration files that are present into the directory `/etc`.

```
1  integrity( $sc1:=".*:ccc_guest_r:.*", $sc2:=".*:.*:.*etc_t" );
2  15$3 : ccc_guest_u:ccc_guest_r:chfn_t −( file { write } )−> system_u:
       object_r:etc_t
```

The following extract of the SELinux policy shows that the guest role can execute the `chfn` application. Moreover, `chfn` as the setuid bit set which enables guest to get the admin role

when executing `chfn`. That example shows that such scenarii of attacks are very easy to carry out. PIGA can efficiently protect against them.

```
1        role ccc_guest_r types chfn_t;
2        allow ccc_guest_t chfn_t:process transition;
3
4   −rws−−x−−x root root /usr/bin/chfn
```

Regarding the integrity of the guest role, the line 1 of the following listing shows a first activity that enables the `dbus` process to transit to the guest role and then transit to the password type. Then, the line 2 shows that a process with the password type can write the `/etc` files and thus modify the passwords. PIGA protects against the combination of those two activities. Thus, if an exploit on `dbus` enables to transit to the guest role, the attempt to then write into `/etc` will fail.

```
1   16!1$11 : ccc_guest_u:ccc_guest_r:ccc_guest_dbusd_t −( process {
          transition } )−> ccc_guest_u:ccc_guest_r:ccc_guest_t ;
          ccc_guest_u:ccc_guest_r:ccc_guest_t −( process { transition } )
          −> ccc_guest_u:ccc_guest_r:passwd_t
2   16!0$11 : ccc_guest_u:ccc_guest_r:passwd_t −( file { write } )−>
          system_u:object_r:etc_t
```

VI. Security Properties Enforcement

That sections adresses the second objective of that paper i.e. how the enumerated illegal activities can be reused to guarantee the requested security properties. It describes the PIGA MAC approach enabling to guarantee all the security properties expressed using our SPL language.

A. Implementation of the PIGA MAC protection

As described in Figure 4, our MAC protection model is divided into two stages. First, the security administrator defines a set of security properties. Generally, the administrator reuses and configures existing canevas such as the various properties proposed in this paper. However, he can also use our SPL language to define dedicated security properties. Then, he uses the PIGA-CC compiler that compares the requested security properties against a mandatory policy (SELinux or grsecurity) in order to compute all the illegal activities existing in that SELinux policy. The set of forbidden activities is then compressed and stored in a database of patterns.

Second, the security administrator can use our PIGA-SP MAC solution in order to ask the enforcement of these security properties by the operating system. PIGA-SP uses a combination of a kernel module PIGA-KM and userland application PIGA-UM. The kernel module hooks the system calls and sends the corresponding traces to the userland application. Each system call is thus suspended and the kernel module waits for an authorization or a deny response. The userland application computes and verifies that the system call corresponds to an activity available in the database of patterns. Next, PIGA-UM allows or denies the considered system call aiming at preventing the occurrence of the forbidden activities. Thus, the system call fails if its execution could lead to the violation of the security properties.

Figure 6 shows the process that allows a target operating system, in our case Linux, to enforce the security properties. Extension of the classical protections is proposed. First, the kernel computes the classical Discretionary Access Control. Second, LSM (Linux Security Module) hooks are applied. LSM enables to stack several protection mechanisms such as SELinux or SMACK. In our approach SELinux is processed before running PIGA-SP. SELinux uses the mandatory security policy in order to allow or deny the system call. In our solution, a system call has to be allowed regarding wth the DAC policy, the SELinux policy, and also the requested PIGA Security Properties.

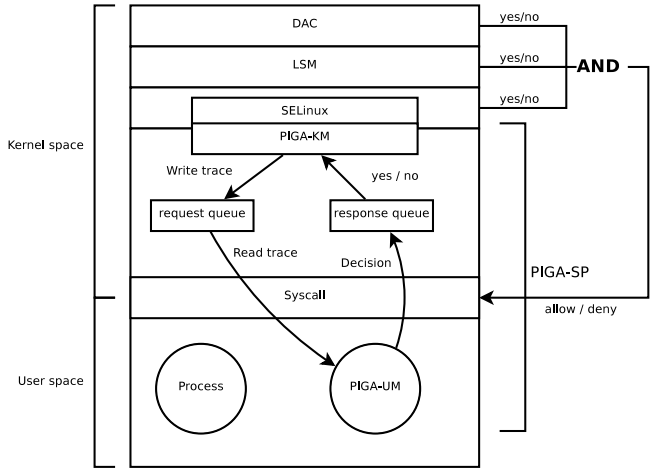


Fig. 6. Process to ensure the security properties at the Operating System layer

PIGA-KM, the kernel part, analyzes all the relevant informations for the considered system call (who made it, from where, what is its type, etc...) in order to generate a trace (i.e. a string) describing the current system call request. PIGA-KM sends that trace to a $request\ queue$ in direction to the userland application, PIGA-UM. The system call is pending, while PIGA-KM is waiting for a message from the $response\ queue$. PIGA-UM computes the response and writes it in the $response\ queue$. When PIGA-KM gets the response from the queue, it sends a decision back to SELinux. The final decision is a logical AND between the PIGA-SP, SELinux and DAC decisions. That final decision allows or denies the system call execution. If allowed, the kernel runs the system call.

The $request\ queue$ is a sequence file in the proc file system. In practice, this file allows to pass a request from the kernel space to the userspace. Once into the userspace, the userland application PIGA-UM reads the request and reconstructs the activities associated to that system call. If a reconstructed activity matches with some available patterns (generated at the compilation stage), PIGA-UM takes the decision to deny that system call.

B. Example of Security Property Enforcement

This section describes how PIGA-SP prevents against real scenarii of attacks. In this example, an administrator uses a ssh connection with the ccc_admin_r role. In this role, he can copy a critical file like $/etc/shadow$ into a file in $/tmp$. This copy could be intentional or produced by an malware

downloaded an executed by the administrator. Next, a user can connect to the system with the ccc_guest_r and read the file, copied by the administrator, in $/tmp$.

Theses actions represents a violation of the following security property that prevents information flows from the ccc_admin_r to the ccc_guest_r role:

```
1  confidentiality( $sc1:=".*:ccc_admin_r:.*", $sc2:=".*:ccc_guest_r:.*" );
```

The result of the analysis of the SELinux policy by PIGA-CC indicates that this violation is possible if we only consider the SELinux policy. This violation corresponds to the following activity:

```
1  3$366122 : ccc_admin_u:ccc_admin_r:ccc_admin_t −( file { write } )
       −> system_u:object_r:tmp_t ; ccc_guest_u:ccc_guest_r:
       ccc_guest_t −( file { read } )−> system_u:object_r:tmp_t
```

The first part of the attack could be simulate by a connexion with the ccc_admin_r following by the copy of $/etc/shadow$ into $/tmp/test$ and a modification of the permissions of the created file:

```
1  # ssh connexion with ccc_admin_u:ccc_admin_r:ccc_admin_t
2  $cat /etc/shadow >> /tmp/test
3  $chmod 777 /tmp/test
```

At the kernel level, the copy of the $/etc/shadow$ file involves a LSM hook specifying that an information has been written by the administrator into a temporally file:

```
1  dec 22 11:23:21 pigaos kernel: type=1400 audit(1277198601.563:1560):
       avc: granted { write } for pid=2056 comm="cat" ppid=1988 dev=
       sda3 ino=534412 scontext= ccc_admin_u:ccc_admin_r:
       ccc_admin_t tcontext= system_u:object_r:tmp_t tclass=file
```

This interaction is allowed by SELinux and also by PIGA-SP because it does not represent a violation of a security property.

Next, a user connects to the same host with the ccc_guest_r role. This user tries to read the content of the file created by the administrator:

```
1  # ssh connexion with ccc_guest_u:ccc_guest_r:ccc_guest_t
2  $ cat /tmp/test
3  cat: /tmp/test: Permission denied
```

This interaction generates the following trace at SELinux Level:

```
1  dec 22 11:25:45 pigaos kernel: type=1400 audit(1277199254.272:1732):
       avc: granted { read } for pid=2080 comm="cat" ppid=1876 dev=
       sda3 ino=534412 scontext= ccc_guest_u:ccc_guest_r:
       ccc_guest_t tcontext= system_u:object_r:tmp_t tclass=file
```

The read interaction of a temporally file by a user in the ccc_guest_r is allowed by SELinux. PIGA-SP prevents this violation because this interaction corresponds to a security property violation.

The trace generated by PIGA-SP indicates the number of the SELinux interaction denied and the corresponding forbidden activity:

```
1  dec 22 11:25:45 pigaos kernel: 3$366122 operation 1732 denied:
       ccc_admin_u:ccc_admin_r:ccc_admin_t > system_u:object_r:
       tmp_t > ccc_guest_u:ccc_guest_r:ccc_guest_t
```

This example illustrates a simple case of security property enforcement. Even if individuals interactions are allowed by SELinux, PIGA-SP is able to control indirect flows or a combination of these indirect flows. Thus, PIGA-SP can guarantee advanced security properties preventing against sophisticated scenarii of attacks including 0-Day attacks.

ACKNOWLEDGMENT

We would like to give special thanks to Jonathan Rouzaud-Cornabas for his participation in the development of PIGA-KM and Maxime Fonda for the development of the SSH confinement with chroot and SELinux.

VII. CONCLUSION

This paper presents two solutions based on SELinux to protect a shared HPC cluster. The first one deals with chroot to confine the user but the approach prevents the user from sharing easily data. The second one is based on two SSH server ports and enables user to share data. This paper focus on the difficulties to prevent sophisticated attacks, using for example several indirect flows, that SELinux cannot control.

That paper shows the efficiency of using SELinux plus PIGA in order to find the illegal activities into the proposed SELinux policy. The found illegal activities can be used to improve the SELinux protection with our PIGA MAC approach in order to better guarantee the confidentiality or the integrity of a shared cluster. PIGA MAC prevents against all the risks of the SELinux policy regarding the various security properties expressed using our Security Property Language. For that purpose, PIGA MAC reuses all the precomputed illegal activities in order to guarantee the required confidentiality and integrity properties. In contrast with SELinux, indirect illegal activities are controlled, permitting thus the prevention of sophisticated attacks.

PIGA MAC can be seen as an advanced HIPS guaranteeing that a system call, terminating an indirect illegal activity, will fail. The PIGA approach can be used also as an HIDS to detect the violations. It is better to use the HIDS approach for properties that correspond to auditing facilities. Moreover, the HIDS approach could be computed on a dedicated cluster. Thus, the impact on the HPC cluster performances is limited. However, PIGA provides a very efficient HIPS approach. Cluster experimentations show that SELinux adds 10% of processor overhead, while PIGA also adds an overhead of 10%. It is a very low overhead for the considered protection that goes much further than the related works. That paper details the implementation of PIGA MAC and gives examples of real scenarii that are blocked by our HIPS.

Finally, the proposed protection enables the real sharing of an HPC cluster while guaranteeing confidentiality and integrity for the partners. Future works deal with the automation of the definition of efficient security properties for the sharing of an HPC cluster. The major difficulty is to adjust the security properties in order to make a good balance between the protection and the usability of the shared cluster.

REFERENCES

[1] M. Blanc, J. Briffaut, T. Coullet, M. Fonda, and C. Toinard, "Protection of a shared hpc cluster," in *Proceedings of the 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies*, ser. SECURWARE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 273–279. [Online]. Available: http://dx.doi.org/10.1109/SECURWARE.2010.51

[2] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM*, vol. 19, no. 8, pp. 461–471, 1976.

[3] J. Briffaut, J.-F. Lalande, C. Toinard, and M. Blanc, "Enforcement of security properties for dynamic mac policies (best paper award)," in *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*, ser. SECURWARE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 114–120. [Online]. Available: http://dx.doi.org/10.1109/SECURWARE.2009.25

[4] W. Yurcik, G. A. Koenig, X. Meng, and J. Greenseid, "Cluster security as a unique problem with emergent properties: Issues and techniques," *5th LCI International Conference on Linux Clusters*, May 2004.

[5] P. henning Kamp and R. N. M. Watson, "Jails: Confining the omnipotent root," in *In Proc. 2nd Intl. SANE Conference*, 2000.

[6] F. L. Camargos and B. des Ligneris, "Automated oscar testing with linux-vservers," in *HPCS '05: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 347–352.

[7] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières, "Making information flow explicit in histar," in *OSDI '06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 19–19.

[8] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris, "Information flow control for standard os abstractions," vol. 41, no. 6. New York, NY, USA: ACM, 2007, pp. 321–334.

[9] P. Efstathopoulos and E. Kohler, "Manageable fine-grained information flow," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 4, pp. 301–313, 2008.

[10] M. L. Damiani, C. Silvestri, and E. Bertino, "Hierarchical domains for decentralized administration of spatially-aware rbac systems," in *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 153–160.

[11] L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozabadi, and O. Mulmo, "Policy administration control and delegation using xacml and delegent," in *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 49–54.

[12] Dan Walsh, "Confining the User with SELinux," http://danwalsh.livejournal.com/10461.html, 2007.

[13] J. Briffaut, J.-F. Lalande, and C. Toinard, "Formalization of security properties: enforcement for mac operating systems and verification of dynamic mac policies," *International journal on advances in security*, pp. 325–343, 2010.

Tailored Concepts for Software Integrity Protection in Mobile Networks

Trust Management to Protect Software for Mobile Network Elements

Manfred Schäfer, Wolf-Dietrich Moeller
NSN CTO - Security Research
Nokia Siemens Networks GmbH & Co. KG
Munich, Germany
e-mail: manfred.schaefer@nsn.com, wolf-dietrich.moeller@nsn.com

Abstract—This paper presents research results on SW security for network elements. Our investigations contribute to the ongoing ASMONIA project, which is focusing on collaborative approaches and on protection and warning mechanisms in 4G networks. This work is dedicated to examine specific aspects thereof, concentrating on software integrity protection (SWIP) to securely manage SW products in mobile networks. Based on an analysis of 3GPP standardization requirements and of existing approaches for integrity protection, solution concepts are proposed and discussed to meet the identified needs. These aim at harmonized approaches for a number of different use cases. Particular account is taken of keeping infrastructure efforts as small as possible, both in operator network as well as in manufacturer domain. The proposed solutions are targeting improvements to integrate and establish efficient trust mechanisms into mobile network elements and management systems.

Keywords-Software integrity protection; secure execution environment; code signing; trust management; Evolved Packet System (EPS); autonomous validation;

I. INTRODUCTION

Software (SW) security assurance has many facets, spread over the entire product life cycle. It has to prevent attacks, arising from maliciously modified SW and associated data, determining a product's behavior.

In the following, the term *SW* may include executable code as well as any configuration information, scripts, data, or meta-data that might be protected together with the SW. Roughly we could split SW security issues into two huge areas, namely (1) to specify and to create a SW product so that it matches given security policies and (2) to assure that in a target system only original SW can be used. The former demands a series of secure SW development processes (which are not further discussed here) and organizational efforts, assuring that SW is free of conceptual flaws, vulnerabilities, and back-doors. The latter is to assure that *after SW creation* unwanted modifications (be it by hostile intent or inadvertently) are prevented or at least will be detected. We concentrate on this aspect also including measures to provide trustworthy hardware (HW) and SW co-design solutions.

Depending on contracts for commercial products SW manufacturers are liable for the SW quality and potentially also for damages and incidents arising from (avoidable) security leaks. Apart from negative impacts of incidents on a manufacturers brand and on customer satisfaction there is imperative need for identification and removal of such flaws, for mitigation and for recovery. Altogether this requires trustworthy SW management and protection.

Focusing on products for mobile access and core networks, we give an insight into balanced strategies on SW protection measures that on the one hand are required by mobile network standardization and on the other hand generally ought to be applied to assure product reliability and trustworthiness as well as to protect SW assets. This publication details the aspects addressed in earlier work [1], providing more room for discussion of requirements and of existing and proposed solutions. Starting with a requirements analysis and examining existing approaches in this paper innovative concepts are derived that beneficially enable to apply the same security infrastructure (in manufacturer domain) to different use cases for SW integrity protection (see Section II.F), while efforts in operator domain can be kept on minimal level.

While many of the strategies and principles addressed by our research work may also be applicable to User Equipment (UE) this is not targeted in *this* paper. But, note that our contribution is closely related to and further supported by the German BMBF sponsored ASMONIA [14] project, where a wider context is envisaged. The project is focusing on collaborative protection and warning systems for 4G networks. In addition to the network centric view as presented in this paper, (among other issues) SW integrity protection for mobile user equipment also will be researched by the consortium, targeting the needs, capabilities, implementation, and integration aspects of mobile phones.

II. ANALYSIS OF SW INTEGRITY PROTECTION NEEDS IN MOBILE NETWORKS

We first examine related security requirements in mobile networks as stated by 3GPP standardization and then we derive more general security requirements, based on an analysis of extended aspects for integrity protection.

A. Requirements related to 3GPP Standardization

In evolution of 3GPP (3rd Generation Partnership Project) [11] standards the upcoming security architectures strictly demand local security capabilities.

In particular, in EPS (Evolved Packet System, see Figure 1, for an example) context requirements are stated for *secure execution-environments* (specific for trusted parts of eNodeBs (eNB) [2]) or *trusted-environments* (TrE, specific for Home-eNodeBs (HeNB) [3]). These arise due to the nature of the EPS security architecture (which, e.g., implies terminating of security relations between user equipment and network and storing of session and authentication keys in EPS base stations) and the attack-prone exposition of such base stations also in public areas, outside the security domain(s) of an operator.

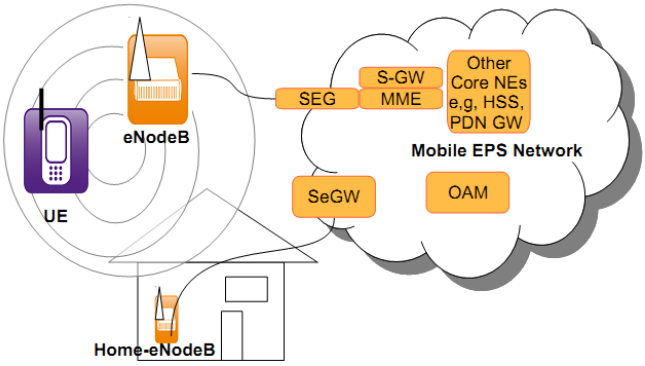


Figure 1. 3GPP EPS Architecture (partial view)

3GPP security requirements include demands for SW integrity checks, e.g., to be applied during secure boot processes whereas any room is left for realization alternatives. As related solutions (if mandatory) have to be implemented in future network products, there is urgent need to identify and to develop efficient methods for trust establishment and management. Essentially, these go back to reliable mechanisms for measuring, for verification, and for enforcement of associated directives for SW installation, loading, and usage.

Specifically, regarding SW integrity [2] demands that '*The eNB shall use authorized data/software*', '*Integrity protection of software transfer towards the eNB shall be ensured*' (clause 5.3.2, for eNB setup and configuration) and regarding the secure environment definition (clause 5.3.5) that '*The secure environment shall support the execution of sensitive parts of the boot process*', '*The secure environment's integrity shall be assured*', and '*Only authorised access shall be granted to the secure environment, i.e. to data stored and used within, and to functions executed within*'. Obviously, authorizing access to an execution environment denotes that any SW, which is brought into it and launched for execution must be targeted to an eNB and must come from an authorized source, which implies proof of origin. Typically, this involves trustworthy boot processes, each time a eNB is started, but also applies to any SW update that has to be made during the life-cycle of such product.

When looking into security requirements for HeNBs, we find in [3] related statements (in clause 5.1.2), explicitly demanding '*The TrE shall be built from an irremovable, HW-based root of trust by way of a secure boot process...*', which '*shall include checks of the integrity of the TrE performed by the root of trust. Only successfully verified components shall be loaded or started..*' and '*shall proceed to verify other components of the H(e)NB (e.g., operating system and further programs) that are necessary for trusted operation of the H(e)NB*'.

Moreover, it is required that the HeNB is enabled to act autonomously as it is stated with '*The integrity of a component is verified by comparing the result of a measurement ... to the trusted reference value. If these values agree, the component is successfully verified and can be started*' and thus needs to be securely provisioned with trusted reference values, as e.g., expressed with '*The TrE shall securely store all trusted reference values at all times*' and '*The TrE shall detect un-authorized modifications of the trusted reference values*'. Further, according to clauses 7.1 and 6.1 in [3], a HeNB must support autonomous validation methods '*If the device integrity check according to clause 6.1 failed, the TrE shall not give access to the sensitive functions using the private key needed for H(e)NB device authentication with the SeGW*', preventing that a malicious device (by self-check) anyhow can connect to the mobile network.

As any trust is based on self-validation processes (which implicitly may also apply for the eNB), very high security expectations are seen for any implementation thereof.

B. *Existing methods for SW integrity Protection*

In the following, we examine available approaches to support SW integrity protection and identify weak aspects and open issues from a mobile network point of view.

1) *TPM based boot control*

Existing methods for usual IT systems, such as known with TCG (Trusted Computing Group) standards (PC-trustworthiness with local ownership concept) cannot be converted easily to network elements and to existing 3GPP operator infrastructures. In particular, methods based on TPM (Trusted Platform Module) paradigms [4] have to be considered very carefully. On the one hand a clear, indisputable value of TPMs (or comparable crypto hardware) is that these may provide sufficient protection for storing secrets and for security operations using such secrets. This involves using the built-in crypto algorithms directly and exclusively without requiring external CPU cryptographic operations, e.g., for network element authentication. On the other hand the TPM *attestation* concept and its implementation (TPM as a co-processor) only provide partial security. There are attack-windows before attestation is completed and the TPM is not designed to parry certain physical attacks, e.g., those modifying the CRTM (Core Root of Trust for Measurement) in ROM or manipulating the TPM interface during the boot process. Doing so a skilled local attacker could inject faked PCR (Platform Configuration Register) settings – but at least has to gain access to the TPM command interface in order to control it.

By nature, the attestation approach is lacking autonomy capabilities. Due to missing local reference values for validation, local systems cannot autonomously determine and take decisions on authenticity and integrity of any SW loaded and measured during boot. In addition, particular account needs to be taken to the fact that managing attestation values over an entire SW product life cycle and for many different products is a challenge in its own.

Moreover, when exploiting extended TPM security capabilities - such as sealing - this imposes a lot of SW and trust management efforts and infrastructure invests, which are not easy to handle. For instance, re-sealing (e.g., of parts of the

An interesting aspect in this paperboot images or of internal secrets) to a new state would require individual provisioning per platform (i.e., due to authorization per TPM and tpm-Proof dependency) and could not be deployed independently from a target platform's security settings.

When looking to 3GPP standardization so far there are no discussions and indications of TPM integration into a mobile network environment. Such implementation specific properties and manufacturer restrictions could hardly be justified and would imply technology-dependent solutions. In best case it is imaginable that for a few very specific network elements such impacts could be accepted but in no case as a template for a broader scope.

As a consequence, integration of TPM/attestation based integrity protection may require remarkable proprietary changes and efforts in the infrastructure, which are difficult to motivate and to sell - apart from the fact that establishment of necessary extensions and provisioning of trust management information needs to be solved by convincing technological means. In addition, regarding implementation the required changes in existing HW (embedded platforms, boards, ASICs) have to be balanced with other design, performance, and cost criteria. Often such trade-offs render it quite difficult or even impossible to simply implant commercial off-the-shelf (COTS) TPM chips into a complex and highly specialized HW / SW platform, which is mainly tailored to meet feature-requirements while security efforts may be capped by defined cost margins.

2) MTM based boot control

In 2004, the TCG initiated the Mobile Phone Working Group (MPWG) to meet use cases and requirements of mobile phones. Based on TPM principles MTM (Mobile Trusted Module) specifications have been elaborated and made publicly available [7], [8], and are clearly in scope of mobile phone industries [12]. In contrast to TPM, the MTM is not explicitly meant as a separate chip specification, rather than it leaves room for different implementations, also as firmware or even as protected SW. The MTM concept can be built on a subset of TPM functionality, but comes with own mechanisms for trustworthy boot.

An interesting aspect in this paper is to examine how and what MTM ideas could be transferred to network elements and how these could be extended. Advantageously, the MTM allows remote management of authorized SW updates by introducing new governance schemes relying on several new types of certificates. As a newness, when compared to TPM principles, the certificate based control (to only execute mandatorily signed and verified software) enables a system to autonomously take decisions during the boot process. Due to its supposed attractiveness the MTM concept if further discussed in Section IV.A.

3) SW integrity protection as used for IT systems

Apart from the specifications introduced by the TCG there are several other individual technologies known, developed and widely used by commercial SW publishers as well as by open source communities. In contrast to TCG (which firstly focused on boot-time integrity) earlier approaches mainly concentrated on SW integrity for SW distribution and installation processes. Regarding the applied security management we roughly we can distinguish three different types of approaches: Those relying on cryptographic 'check-sums' (pure hash values as e.g., applied by some open source communities, such as OpenOffice [16]), those using code signatures based on Web-Of-Trust principles (e.g., PGP/GPG based code signing as used with RPM [17]), and those integrating with PKI principles (e.g., as established for JAVA [18] or Symbian Signed [19]).

Concepts based on pure 'check-sums' suffer from the difficulty to obtain valid reference values from trusted sources (no inherent proof of origin) and to reliably store these over a potentially long time – thus, these reference data are always susceptible to man-in-the-middle (MITM) attacks. Moreover, extended security control (e.g., regarding expiry, revocation, self-validation) is rather limited or simply not possible. Note again, that also the TPM paradigm does not natively solve these issues!

Considering mechanisms relying on Web-Of-Trust (WoT) principles, one may complain that WoT based methods do not match very well with demands for vendor driven governance and security control over network elements. As a matter of fact also WoT inherently does not reliably exclude MITM attacks. Everybody could create self-signed signing certificates and keys, as there is no mandatory registration authority established. So, trust always lies in the eyes of a believer. Apart from this deficiency a WoT usually is neither based on enforceable hierarchies nor on expressive and standardized certificates. Moreover, WoT principles do not support effective and reliable revocation schemes as usually there are many trust relations and unclear governance schemes involved (no public policies, no CRLs) and no 'official' mechanisms or entities for enforcement are available.

Of course, for user centric scenarios, individual products, or platforms (e.g., Open Source Linux distributions, based on RPM or similar package management systems) such mechanisms are beneficial. But as we are looking for generic templates for remotely manageable SWIP mechanisms for mobile network equipment, we do not deeply analyze these approaches in this paper. This does not mean that we generally dislike or ignore such concepts, but we have to apply them in the right context and scenarios.

From a vendor's perspective - who should be able to fully control the security capabilities and integrity of its products - potential difficulties may arise from inadequate fundamental security building blocks and in particular from unsuited key and trust management strategies and weak control mechanisms. This clearly argues in favor of PKI based approaches, which are compliant to accepted standards and security best practices and provide well proven governance and control principles (e.g., as defined in X.500 [27] and in particular with X.509 [15]).

While many of the PKI-based known signing concepts (JAVA, Microsoft's Authenticode, Symbian Signed, IBM's Lotus Notes, etc.) apply efficient and partially even comparable mechanisms, they are not directly applicable to the needs of manageable SWIP for mobile network equipment (which usually consists of a number of very different products and technologies). On the one hand such approaches usually are specific for one particular product or technology (e.g., operating system, programming language, controlling sand-box, run-time environment, web- or IT-applications, vendor specific UE-equipment, etc.) and on the other hand they are mostly targeted to support security requirements of distributed developer communities.

In most cases they rely on outsourced PKI entities (certi-

ficate and registration authorities) and on verification components, which often allow importing of arbitrary SW publisher certificates and accept umpteen root certification authorities (CA). If a user or administrator decides that these are trustworthy he can change the trust management settings by local administration. Vice versa, preconfigured trust-anchors and credentials could be removed on user decision. Consequently, in addition to local control, such systems chiefly target to enable tracking and juridical inquiry of malicious attackers, which by hostile intent previously have applied for SW publisher keys and certificates.

Some approaches combine code signing with explicit authorization concepts at application level. For instance, this is realized by different types of certificates (e.g., using 'capabilities' as introduced with Symbian Signed), which are associated with classes of API calls with appropriate scope and privileges. Adherence to such assignments can be checked by signing entities (before issuing a valid signature for an object under test), as well as by the devices themselves during verification or execution. To give more examples, mechanisms used by JAVA or IBM's Lotus Notes [20] control application privileges via sand-boxing approaches, but use local administration to set policies and rules for execution.

C. Run-time Aspects

When looking beyond the scope of boot-time or installation-time integrity checking additional security improvements are needed to provide attack resilience during long-term operation. These have to be faced as many network elements (in particular threatened eNBs, HeNBs) may be booted or updated only rarely. Then they have to be active for weeks or months, whilst the trust in boot-time checks is the more diminished the longer a system is running. Potentially this is caused by attacks occurring during operation, applying both for local, manual manipulations as well as for remote SW attacks, which cannot be prevented by boot protection alone. Consequently, there is urgency for methods and mechanisms assuring SW integrity at run-time at least for critical security operations. Such critical operations (e.g., as needed for key and credential management, for authentication, or for verification processes) require trusted code, which can only run if *before execution* it is proven to be integer and to stem from an authoritative source.

Run-time integrity issues are partly covered by TPM based improvements. For instance, with DRTM (Dynamic Root of Trust for Measurement) mechanisms are known to allow lately measuring and launching SW in a TPM compliant execution environment [4], [5]. The DRTM mechanisms assure that code, which is to be started, is measured properly (e.g., Intel is using authenticated code modules for this purpose [13]) and then executed, but *does not prevent from loading untrusted code*. Such approach requires TPM based attestation (with all the hurdles mentioned above) in combination with dedicated CPUs, which have to support specific instructions and bus cycles. While an external challenger is enabled to prove what has been executed (during run-time) on a DRTM equipped system, the DRTM operation itself is not able to verify any manufacturer code prior to execution. Again we miss an autonomous mechanism enabling a local machine to enforce rejection of manipulated code, preventing execution of any hostile operations at any time. Moreover, the selection of DRTM enabled CPUs may be in conflict with other CPU selection criteria to best match the needs of the specialized embedded architectures of a mobile network element.

The IMA approach [6] is an interesting extension of TPM concepts, introducing TPM protected load time integrity measurements of file-based executables, libraries and data, which are aggregated into a series of TPM signed lists. As with the native TPM principles, IMA relies on attestation paradigms, requiring external entities for validation. Apart from lack of autonomy the major problem of such approach again seems to be the need to maintain a TPM specific infrastructure as well as the efforts to interpret and validate a potentially huge amount of attestation data, which is reported on request. Such data has to be 'known (i.e., must be securely provisioned)' externally or must be re-calculated (where referring to any sequence of loading is not required in the IMA case).

There are other approaches such as Tripwire or Samhain [23] following alternative principles based on file-level integrity checks, which do not rely on a TPM infrastructure. They come with own associated, administrated client/server architectures and self-created, protected databases with 'trusted' hash values for validation. The run-time checks are triggered periodically or based on events, while checking modules are protected at kernel level, which may be sufficient for some attacks scenarios. A particular risk may be the fact that trusted reference values may not (or not only) be created inside a secure developer environment at manufacturer side, but in the operational domain itself, which is not only an organizational, but also a liability issue.

It is worth to mention the DigSig proposal [24] representing a load-time integrity checking solution, which relies on PGP signed ELF binaries (as provided via the Debian BSign utility), but is applicable to Linux systems only (due to dependency on Linux kernel integration and on ELF files and tools). The charm of such approach is the fact that signatures are embedded into ELF binaries, thus no separate data base is required. Moreover, signatures can be created externally, therefore local creation of trustworthy reference values is avoided and the system is enabled to take advantage of the benefits coming along with a code signing approach (e.g., proof of origin, and signature revocation, which is also supported), even though restricted by the WoT paradigm. Such solution is related to previous work [25], also based on signed ELF binaries, relying on comparable principles for run-time integrity protection.

Sand-boxing solutions such as introduced with JAVA cover PKI based SW integrity protection during installation and download scenarios and realize mitigation concepts during run-time, but may be restricted due to an individual programming language environment and due to individual sand-box constraints. Sand-boxing is not only related to integrity checking, but also may constrain program capabilities and performance during execution. This may or may not be a problem, depending on the application, but is limiting general applicability.

For reasons of completeness it should be mentioned that there are also run-time protection methods known, which make use of specific CPU level concepts, such as Intel's System Management Mode (SMM) (e.g., compare [26]). We do not further discuss these in this paper, as they are too dependent from processor capabilities (like the DRTM mech-

anisms, mentioned above) and thus are not ideal for generic templates for SW integrity protection, we are aiming at. Of course, it is well understood that mechanisms making use of low level HW properties can achieve a higher protection level - but usually at cost of flexibility and portability.

D. Autonomy and Remediation Aspects

Autonomous SW integrity protection and trust management mechanisms are highly desirable, enabling a system to take own, reliable decisions e.g., to deny sensitive services, to boot to fail-safe-mode if a new SW release is defective or to generate and transmit (or to store) signed incidence messages in case integrity violations are detected during run-time checks. Autonomy decreases efforts in network and increases security as a system knows about its own trust state, before it connects to a network. Of course, this may be limited to attempted attacks, which can be detected before they are effective and to non-persistent attacks, which can be cleaned, e.g., by re-booting or re-installing, or to attacks, which do not successfully affront and neutralize the integrity protection mechanisms themselves.

Critical are situations where a large number of systems are actually compromised by sudden attacks. For such cases robust remediation mechanisms might be implemented, which are resistant against certain classes of attack, so that they cannot be smarted out in some way - or at least not too easily. Such remediation mechanisms may require reliable, autonomous local mechanisms and even interaction with supporting network entities, assuring that affected systems could be repaired securely from remote. The reasoning behind is that in mobile networks, and in particular with the flat architecture introduced in EPS there are a huge number of systems in field, widely distributed and very often in secluded areas. Any personnel to be sent out for emergency or management services needs time and raises cost and efforts. In some cases, e.g., for HeNBs, it might also be acceptable to involve the hosting party (i.e., the user) into remediation actions, but this depends on the underlying trust model.

In particular, those attacks seem to be very precarious that emerge from remote SW injection attacks occurring during run-time. This is because they could be launched against a large number of systems simultaneously, causing partial outage of large network segments or even complete network breakdown.

Clearly, autonomy and remediation mechanisms require robust implementation, which might by quite expensive and thus, efforts always have to be balanced by cost-efficiency considerations.

E. Generalization

The above considerations may be very specific to 'standardized' requirements for integrity and trustworthiness of exposed network elements such as eNBs and HeNBs. However, the mechanism applied should also be beneficial to defend against attacks that may target or affect elements located in a (more) secure domain. Particularly, this applies if we want to exclude attacks that could be injected via the SW delivery and installation chain. Therefore, for such broader scope an important strategic goal is to re-use SWIP concepts as well as the involved components and infrastructure at the greatest possible extent, while efforts and changes in operator networks should be minimized. Understandably, it is hardly acceptable to apply (too many) different concepts for different products, if this requires operator invests, be it for organizational or operational measures or be it for technical equipment. The ideal case would be that mechanisms for managed SW integrity protection can smoothly be integrated into existing nodes, protocols and do not require unnecessary changes in standardization.

Consequently we aim at generalized and harmonized approaches for managed SW integrity protection. Such solutions shall provide adequate security and shall be suited to protect many other SW products in a mobile network (also outside the scope of EPS), widely independent from architectural aspects and from complex implementation details.

When thinking of SWIP for products in core network (i.e., those residing in the security domain of a mobile operator) essentially we can concentrate on intended SW update and SW delivery interfaces and processes, as mainly these may offer chances for malicious intervention. On the other hand, physical protection and tricky implementation issues against local attacks may be of less importance there.

Complementing the above analysis, particularly the following requirements are relevant for generalization:

- Ensure that SW (that may be composed of different components and data) has not been altered after creation process. This includes accidentally infected SW as well as *any* intentionally modified code, inserted into the SW update, maintenance or delivery path.
- Identify that SW (and associated data) is coming from a specific, authorized source (Proof of Origin).
- Verify that code is trustworthy and authorized for a specific purpose or target system. This may be expressed implicitly (by SW package) or by explicit verification of meta-data or attributes.
- Allow associating SW with unmodifiable directives and privileges for code, memory and data usage, according to the claims of an authoritative source.
- Support 'static' (before run-time) as well as 'dynamic' (during run-time) protection, preferably based on the same (cryptographic) measures and mechanisms.

F. Holistic View and Intended Use Cases for SWIP

Extending the conception of generalization a visionary idea of SW integrity protection is shown in the Figure 2 below. This holistic view reflects how SW may be used in different execution environments and in different operational stages, starting with SW creation and delivery processes and

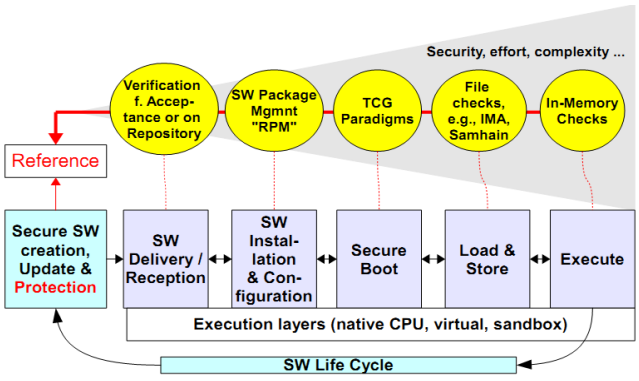


Figure 2. Holistic view on SWIP: SW in different operational stages

then passing through the possible usage modes and life-cycles.

The major use cases for SWIP include SW verification

- after delivery (at point and time of acceptance); this use case is relevant for scenarios where chiefly SW delivery processes need to be protected from attacks against SW, while it is shipped from the manufacturer to customers or to service personnel. In some cases one-time verification may be sufficient and after acceptance further protection is not needed.
- during installation; this use case requires SWIP (verification) integration into a SW installation process, which can performed locally (self-installation) or by a remote installation server. Protecting downloaded SW (which is installed at run-time) or native SW for virtual environments might also be included here. In dependence of requirements for the installed system it may be beneficial to combine SWIP with directives for the installation process, e.g., on versioning or patching, on revocation or on invalidation of previously installed SW.
- during the boot process; this use case (which typically requires active integration into a system's boot architecture) corresponds to mechanisms addressed earlier, when discussing approaches as introduced by TCG standardization bodies. Essential characteristics comprise the sequential dependency of several SW modules being loaded and the local control taken over the boot process.
- while it is stored in a file system, data base or in flash memory; this use case actually corresponds to a run-time verification scenario on storage level, where SW may be continuously verified, be it periodically or be it triggered by events created through actions, which may affect the stored data. Typically, such scenario may be effective if the status of an installed system must be checked over a long time, and may be seen as a completion to the SW installation use case. Note that the SW verified in the file system may either be in use currently or not, or it may even be stored in a repository.
- while it is executed in cache or CPU memory; this use case again is a run-time scenario. In contrast to the preceding use case, only active SW (i.e., such SW which has been loaded into memory) is under examination. In practice, this use case is the most challenging one and many efforts must be spent for efficient implementation.

In all cases, SWIP aims at checking whether SW (i.e., invariant parts of it, such as executable code or initial data) in each operational stage has been modified, when compared to the originally created reference SW. Depending on expectations on attack resilience, efforts and methodological complexity may be very different.

While partially well known or even standardized individual methods for different aspects of integrity protection are available, in some areas this is still requiring fundamental research. Particularly, SWIP is the more challenging the more we aim to inspect a system during (long) execution time and the deeper we look into a system's CPU memory space. However, at the same time the achievable security and trust-level will remarkably increase when moving from a static view on system integrity towards a dynamic one (i.e., SW module loading and execution). In the context of mobile networks the latter may become of significant importance, regarding indispensable long-lasting trustworthy operation of systems in field (e.g., operating several months per boot).

As conditions of target systems and SW environments are varying, actually a huge number of product specific solutions is required, in particular when confronted with the HW and SW particularities of our systems (e.g., SW installation and update processes, run-time environments and operating systems) where the SW is verified and used.

Consequently, it is not surprising that currently a harmonized, integrative approach is missing, which could cover all the use cases above with a unified or adaptive method. Nevertheless, this would be very beneficial and from the beginning we should aim at unification and adaptability of methods *as far as possible* and this particularly requires identifying those aspects which are widely independent from platform or implementation specific solutions.

The guiding principle of our approach is the cognition that by applying certificate based SW signing schemes (the manufacturer's) infrastructure efforts could be harmonized, while we still have to accept remarkable differences for system specific implementation and secure anchoring of trust and verification mechanisms. Such infrastructure involves, e.g., managing certificates, PKI extensions, signing mechanisms and entities, certificate policy guidelines and rules, key management principles, approval work-flows, secure SW development processes, data structures, conceptual templates, common verification and measurement tools, and so on.

As this all could be provided by the manufacturer and to a large extent could be driven by the products themselves or by (product specific) network management components, impacts on an operator's infrastructure could be kept minimal, e.g., limited to manageable changes in existing mobile network equipment.

III. SWIP PROCESSES AND TARGET SYSTEMS

In the following, we will propose and discuss strategies and concepts to match the requirements and visions as introduced above. Firstly, we consider processes as relevant for SWIP and secondly, on a conceptual level (i.e., without reference to concrete network elements) we examine influences of SWIP on target systems in the network environment.

A. SWIP Processes

For SW integrity the following four processes are essential and have to be realized for all the use cases mentioned:

(i) The *protection* process where the SW becomes 'integrity protected', e.g., by applying cryptographic methods;
(ii) the *verification* process where it is checked (verified) whether the protection has been broken or not;
(iii) the *enforcement* process where the SW is securely stored, distributed, installed, or executed, following instructions that may be part of the protection paradigms;
(iv) *infrastructure processes*, which are required to enable and support the others listed above.

Ideally, protection (i) is applied as early as possible (i.e., directly after SW is created, tested, and released, e.g., in the build environment). Verification (ii) and enforcement (iii) are done as late as possible (i.e., just before the SW is used or executed) and even better continuously as long as the SW is installed (or is running). It is evident that these processes are closely related to each other and must follow common mechanisms and paradigms that may require information exchange among each other (e.g., keys for encryption or signing mechanism or trusted reference parameters for hash values). Preferably (for a manufacturer dominated approach) the process (i) is executed in a secured domain at vendor side, while the processes (ii) and (iii) are executed in the operator network, but based on manufacturer-provided SW, key material, credentials, and mechanisms implemented within network elements. There may also be other constellations (e.g., where a system itself is responsible to run local protecting processes (i)), but these are not discussed in this paper.

In addition to the above processes, preparatory and operative infrastructure support and management processes (iv) are required, in particular to establish PKI and signing components and to control key material and credentials (in case SWIP is based on certificates) or to provide reliable reference values and trusted sources and secure management and validation capabilities for these (if SWIP is based on pure hash values or attestation principles).

Regarding harmonization certainly the focus lies on infrastructure impacts, but also the above processes (i, ii, iii) would profit from a common methodological framework, as involved tools and data structures to a large extent could be made similar and adaptive.

B. *Target Systems*

We define a target system (TS) as the 'consuming endpoint' (the platform for which the SW is designed and which hosts the execution environment where the SW is running).

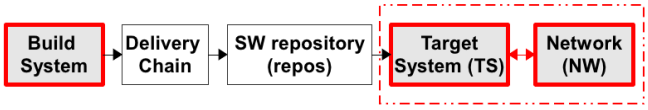


Figure 3. SWIP-aware target system, verified by network

Figure 3 shows a SWIP system where (ii), (iii), and partially (iv) are shared between the TS and an extra, external node, e.g., a verification server residing in the network (NW). Into this category fall systems that

- implement *trusted boot*, following attestation principles and TPM technology (both based on CRTM or on DRTM);
- realize *Integrity Measurement Architecture* (IMA) [6], a load-time extension using TPM attestation principles;
- act as monitoring systems interacting with network, such as Tripwire, Afick, Samhain or also IMA;
- follow principles as applied with TCG's trusted network connect (TNC) [9].

As explained such use cases (when based on external validation) may impose remarkable difficulties – regarding applied security paradigms and trust managements –, which are costly to manage in a mobile network environment. Even if these are not seen as our preferred solutions, some principles could be used, if appropriate.

In Figure 4 the ideal case is shown, where (ii) and (iii) to the greatest possible extent are assigned to the TS. This would be the best solution regarding effort minimization for the network (also regarding (iv) for setup and provisioning). This category includes the following use cases for self-subsistent SWIP-aware TSs, which are enabled to autonomously

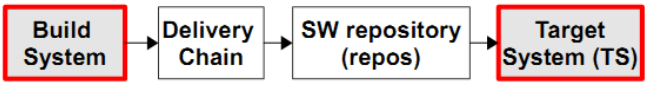


Figure 4. SWIP-aware autonomous target system

- implement *secure boot*, doing verification and enforcement during the start-up process, e.g., as introduced by the Mobile Trusted Module (MTM) specification, issued by TCG [7], [8], see Section IV;
- verify and enforce SW integrity at installation-time, every time before a SW component is installed or stored into a local SW base. Typically, this can be integrated in installation systems, such as packet managers;
- verify and enforce SW integrity, each time a SW component is loaded into system memory and then executed;
- self-monitor and verify SW while a system is running, triggered periodically or by system events (e.g., file access, socket activity, system call). Both, memory-images or files could be checked by such monitoring process;
- … and only occasionally need additional support from SW-repository (or an OAM server) for individual cases, e.g., for autonomous SWIP related SW update processes, for remediation or security management (e.g., remote exchange of secrets, credentials, or of trust anchors).

Considering generalization also SWIP-unaware TS (see Figure 5) are of interest, i.e., those where processes (ii), (iii), and (iv) are completely treated outside a TS. SWIP then is concentrated in network entities (such as operator side repositories *o-repos*, e.g., an OAM or SW management system) and the TS systems security architecture remains unaffected. It is essential that there must be a strict trust relation between the o-repos and the TS, which simply plays a passive role for SWIP.

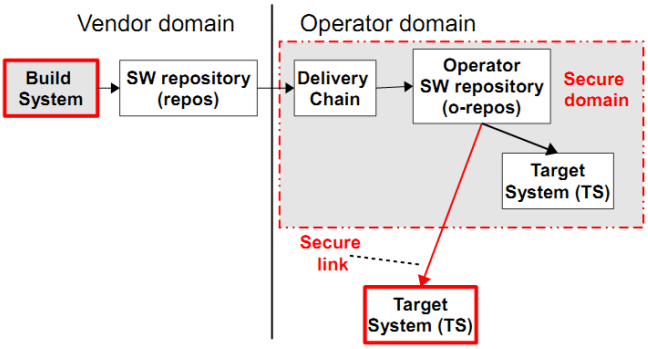


Figure 5. SWIP-unaware target system, supported by network

Such unaware TSs cannot protect themselves and must fully rely on secure domains and on the network entities they

are connected to. It is obvious that such solutions cannot be applied in insecure domains, unless some basic security, such as for secure communication of raw data or SW is provided (here the major use case is to protect the regular SW delivery and update processes, managed by NW entities, like *o-repos*).

It is evident that for the latter case, some of the targeted use cases cannot be applied, but this might be acceptable in accordance with the risk assessment and cost considerations of involved network elements.

IV. CERTIFICATE BASED APPROACHES

As already implied earlier we are convinced that a framework based on certificates and PKI entities would be most suited to fulfill the requirements of managed SWIP for mobile network equipment. In addition to autonomy aspects (as explained in Sections II.D and III.B), we expect positive effects for generalization and harmonization (see Sections II.G and II.F) as well as for vendor dominated governance principles and the expectations we may have on security, regarding the use cases and product-life cycle aspects as introduced.

As an example in the following we discuss an existing approach and make proposals for further improvements, be it from methodological point of view, be it for implementation.

A. The MTM approach

As already mentioned in Section II.B an inspiring idea has been proposed by TCG to assure trustworthiness for mobile phones. Aligned with (basic) TPM paradigms the MTM specification defines certificate based mechanisms for verifying and running trusted software on mobile phones. The new idea behind the MTM specification is to support *secure boot,* allowing local verification (ii) and enforcement (iii) during the boot process, which again may involve several mutually dependent modules (i.e., to be loaded sequentially). MTM introduces so-called RIM (Reference Integrity Metric) certificates containing integrity measures and references to public keys (assigned to so-called RIM_Auths), to verify a complex certificate chain against a (e.g., built-in) root verification key. According to this, the MTM specification enables a system to act autonomously, particularly to identify and to verify downloaded SW, to perform proof of origin and to take decisions in case of detected integrity violations.

As implied, MTMs can be built upon the TPM architecture, but only need a subset of the TPM functionality. As RIM certificates integrate measurement values (as specified with TPM) - in addition to *secure boot* mechanisms - attestation protocols still can be applied, involving external entities if needed.

Regarding SWIP there are many correlations between the requirements for a mobile phone and managed SW integrity protection for NEs within a network infrastructure as demanded above. The certificate based integrity protection principles of the MTM specification can be exploited and beneficially be applied in the context of SWIP strategies and related trust concepts in a mobile network. Such ideas perfectly harmonize with the autonomous and generalized use cases as depicted in Figure 4 and Figure 5, while management support in network infrastructure can be kept at a minimal level (certificates are self-describing and attestation might not stringently be required).

As further explained SWIP based on adapted MTM concepts might very well support both, EPS security needs (as specified with eNB or HeNB), as well as generalization aspects, as explained in Sections II.E-G. In Section IV.B we propose required extensions or adaptations, taking the MTM approach as an exemplary framework. Alternatively, we also could found our concepts on another PKI / certificate based method, but the MTM seems to be a suited start point and might be 'easier' to extend, due to existing ideas on implementation in embedded systems (including TPM mechanisms underneath), to (multi-) vendor centric governance schemes and to solutions for the 'secure boot' use case.

Note that additional local security requirements beyond the scope of SWIP, e.g., related to uniqueness and ‘secure or trusted environment’ (such as secure key management, storage, and usage, and device authentication to prevent HW cloning etc.) must also be fulfilled, but are not described in all details by the solutions below. However, we give some hints on the relevant implementation aspects.

B. Adaptations of the MTM idea

We consider useful adaptations of the initial MTM idea to extend and improve SWIP methods for mobile network elements:

1) Focusing on secure boot

When applying *secure boot,* the additional value of attestation may be rather limited as compared to the organizational efforts and equipment to be invested in network infrastructure. Based on self-validation it must be assured that a system connects to a network only if the boot-time verification was successful. Otherwise, the system shall deny any interaction with the network, except, e.g., for OAM purposes. As a precondition a highly secure root of trust (e.g., non over-writable verification key) must exist. Further secure key-storage (e.g., read and write protection for private authentication keys) and secure usage for such keys in a secure (execution) environment must be guaranteed. The secure boot process is part of the establishment of such a secure environment.

The value of an additional attestation is questionable (if done to reveal a system's trust state during long-time operation), but it has some relevance if we just want to know if a new SW version successfully has been installed. See Section V for alternative approaches, which avoid involving a complete and difficult to manage TPM infrastructure and deployment.

2) Implementation aspects

Just as with TPM any security heavily depends on a secure implementation of a CRTM, in a similar way this applies to the MTM. The initial ‘immutable’ code in the MTM case is called ‘Root of Trust for Verification / Enforcement’ (RTV / RTE). Based on a risk assessment it has to be decided in each case separately which foundation for the security of RTV / RTE and the root verification keys has to be selected. In many cases (e.g., regarding remote SW attacks) it might be sufficient if these data are not over-writable or are only mutable via strong authorization mechanisms that cannot be surmounted via instructions executed by a CPU.

While the specification allows integration of TPM hardware underneath, the MTM concept is also intended for separate firmware or SW implementation. For reasons explained

above, this is of particular interest for systems that cannot simply make use of commercial TPM hardware solutions.

However, in all cases 'sufficient protection' has to be provided for using and managing local secrets and credentials, as well as for sensitive processes (e.g., for local measuring and reporting). Apart from TPM or comparable crypto-HW, simpler ASICs or CPU-level integration are effective to achieve higher security levels against pertinacious attacks. See related proposals in Section V.

Evidently, for some scenarios (e.g., where we do not expect highly motivated and perfectly skilled attackers) it might be sufficient to make use of efficient SW integration techniques like kernel-space protection, system level attack mitigation or virtualization for implementation. In practice, accurate shaping of these mechanisms must be based on an individual threat and risk analysis

However, due to the focus of this paper (which concentrates on the conceptual approach) we do not step into details hereto.

3) Extending certificate concepts and use cases

When thinking of generalization for SWIP, the following modification of the MTM principles is gaining importance: While RIM certificates are perfectly tailored to implement *secure boot*, they are not designed to support the needs of other SWIP use cases (e.g., SW installation, run-time aspects or SW delivery, as well as bundling with extended authorization concepts). A more flexible and adaptive structure instead of RIM certificates (which actually is not a certificate in PKI sense, but 'standardized' signed data for a specific context) is required, which is adaptive to the needs of a specific SWIP use case or to the particularities of a SW product.

In Figure 6 we introduce a *generic Signed Object (SO)* to substitute RIM certificates. SO preferably might be implemented as XML signed objects to gain profit of the power and flexibility of XML and the associated XML signing framework [10], but alternatively, CMS [21] implementations could be taken as well.

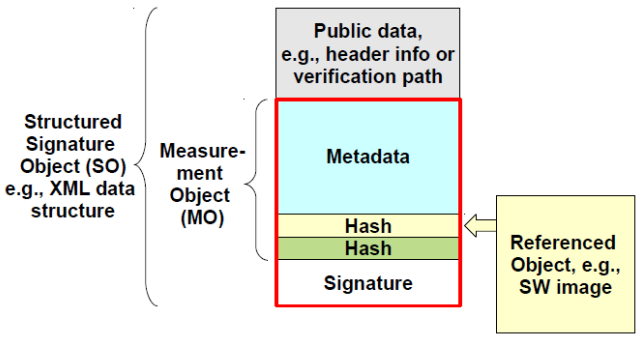


Figure 6. Generic Signed Objects (SO), describing the protection context

Apart from verification information (e.g., intermediate certificates of signing entities) or other public data, a SO consists of one or more signed Measurement Objects (MO), which essentially contain information, which is measured and gathered by the SW protection process (i). Such information may also contain meta-data (descriptors, circumstantiating the MO) and measured objects, which are representatives of referenced objects (e.g., hash values of one or more SW modules). As shown in Figure 7, such MO meta-data may include

- Object descriptors, specifying the measured objects together with references to associated policies.
- Measurement descriptors, specifying the format and syntax of the MO and of MO elements.
- The Measured Objects (MdO) themselves; either this can be hash values of referenced objects (e.g., a SW module or archive) or even embedded data, such as a small script or configuration information. Also, other existing external MdO or MO information might be referenced, supporting a hierarchical approach (e.g., an archive together with individually protected files stored in this archive).
- Entity descriptors, specifying the responsible entities (e.g., company), together with legal implications (e.g., disclaimers or warranty clauses).
- Crypto descriptors, specifying the applied cryptography, e.g., hashing and signing algorithms.
- Policies, which express directives according to claims of the authoritative (signing) source. Policies may include explicit rules for verification and enforcement processes or they may describe general dependencies between SW modules (including compatibility information or rules for 'sequential loading' as used with RIM certificates). Another scope of policies could be expiry or revocation of individual SW packages (which need not necessarily imply revocation of a signing key and the associated certificate). Polices can be static ('do not load module x together with y') or conditional ('if the target platform CPU is ABC, do not load driver Z'), i.e., may depend on information, time, or the state of the system to which they are applied.
- As such SOs are much more flexible and expressive than RIM certificates, they perfectly match with the requirements and visions as stated in Section II. Depending on the meaning of the descriptors and in particular of the associated, static or conditional policies, very different rules can be stated to influence and to control the processes in the SWIP endpoint (e.g., an OAM server) or within a trusted (i.e., verifying and enforcing) TS itself.

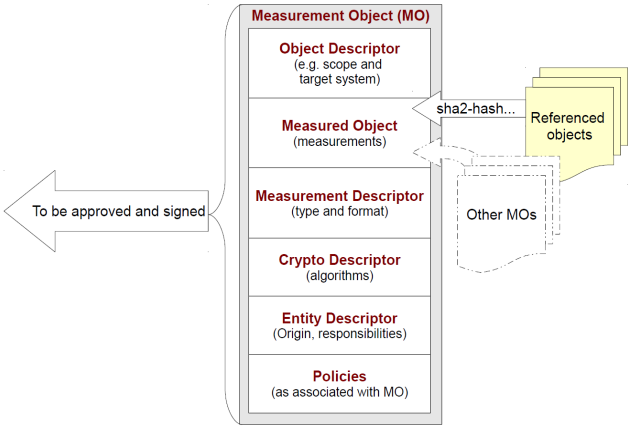


Figure 7. Measurement Objects (MO), specifying measured data

In addition to directives and conditions for SW usage, policies may express directives for the usage of MOs themselves, e.g., by specifying governance rules that have to be

applied for a specific object (such as invalidation, deletion, upgrade, required patches, etc.).

The syntax and semantic of such SO may be associated with a company or with a specific use case or product being managed by an individual responsible party. It should be emphasized again that SOs may cover the full meaning of RIM certificates as a specific sub-case.

4) Governance principles

While the native MTM specification is not mandatorily aligned with X.509 and general PKI principles, we recommend to adapt the MTM governance principles to a (potentially vendor controlled) X.509 compliant PKI infrastructure. This only applies to the so-called RIM_Auths and the upper hierarchy up to the root CA. It should be noted that this part of the MTM specification could be easily integrated into X.509 elements and could be adapted to be governed via specific PKI policies, according to the needs of an individual manufacturer. The MTM specifications mention this, but do not specify any details.

V. HW LEVEL TRUST IMPROVEMENTS

In the following Sections, we discuss HW level improvements increasing security and flexibility of trusted systems like the ones alluded above.

A. Authorized SW Update

The first method locally enables authorized updating to new versions of protected SW and data that only after successful verification will be written to non-volatile storage, e.g., Flash EEPROM or hard disk.

In real systems run-time attacks enabled by vulnerable SW (e.g., exploits) are likely to happen. However, solutions for boot time protection cannot not directly provide prevention against (later) run-time attacks, which intend to take control over a system and to run with malicious functionality. Certain exploits could even try to prevent reliable and verifiable SW updates of the system, which for the future could leave the system with an old, flawed SW version. This would hold the device in a vulnerable state where the old, vulnerable SW version is still booted during next secure boot, and still accepted as a valid version, even if the new SW version should already be installed. Therefore an unacceptable security leak may arise.

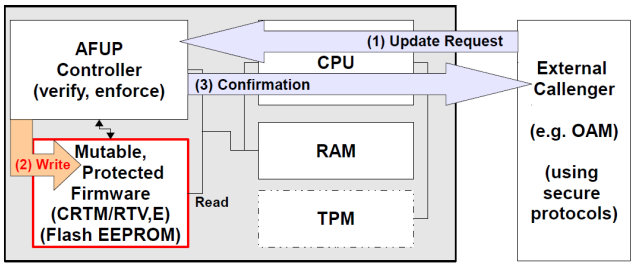


Figure 8. Authorized Flash Update Process (AFUP)

Moreover, it must be prevented that a more sophisticated SW or even local attack could change the content of any persistent trusted (i.e., already verified) code.

For external entities (i.e., regarding secure connection to a network) it is essential that either reliable attestation or one-time proof of a successful secure SW update process can be established.

In the following, we describe a solution that can be established without the need to build up and to maintain an attestation infrastructure and to deal with TPM integration. In addition to authorization and autonomous integrity protection the proposed solution provides a mechanism against specific, persistently implanted or repeated run-time attacks (against required SW updates).

The solution uses Flash EEPROMs protected by an Authorized Flash Update Process (AFUP) depicted in Figure 8, communicating via the system CPU. The control part of this process (the AFUP controller) can be implemented via dedicated hardware (e.g., an ASIC), which by design is the only unit that controls flash programming (at least for critical parts of the flash memory), and could not be affected by defined classes of attacks (e.g., CPU driven SW exploits or even certain physical attacks). For verification it can rely on 'roots of trust' residing in the flash memory.

In its fundamental operation, AFUP uses pre-configured secrets and credentials for a protected communication with an external requester, which initiates the communication by sending an update request (1). The delivered SW (that may also be the CRTM or RTV/RTE SW) is integrity protected (i.e., accompanied by signed objects SO) and is only updated (written into flash memory) upon a successful verification by the AFUP controller (2). On success the AFUP controller sends a confirmation (3) to the external requester which now can be sure that after a next boot the system is updated to an 'invulnerable' SW version and can be trusted again.

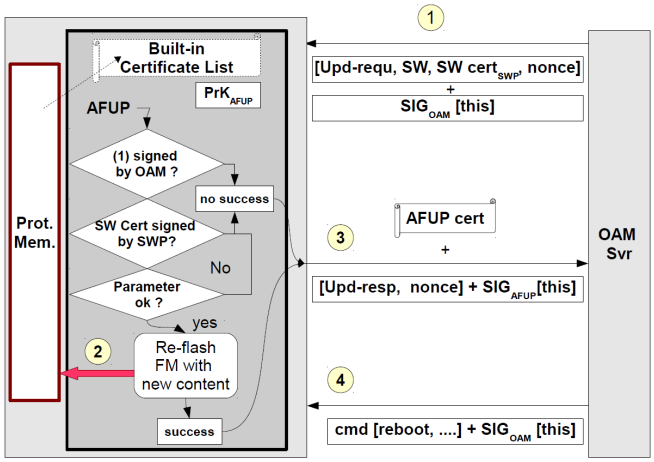


Figure 9. Example for AFUP communication protocol

In Figure 9, an implementation example is shown detailing the AFUP communication relying on a certificate based security protocol. According to this the AFUP controller is personalized with a (well protected) private key and a built-in, write protected certificate list (this list could be stored, e.g., in the protected memory), which initially have to be implanted by a secured process, e.g., during manufacturing. The certificate list might contain the manufacturer's root CA (Certificate Authority) certificate under which certificates for a SW provider are issued, denoted here as SW $Cert_{SWP}$.

In accordance with the scheme shown in Figure 8, the communication is started by a SW update request (1), sent by an OAM server, which typically is located in the operator

network. This request may contain the new SW itself or may also provide a link to a location where it can be fetched from. The SW itself is protected by a signature, issued by the SW provider SWP and the associated certificate, which -in this example- is part of the update message .

To prevent from replay attacks and to assure a trusted source the OAM server adds a nonce and signs the message, which can be verified by a root certificate, which is stored in the AFUP's certificate list depending on the key material the OAM server is provisioned with. In the simplest case this could be the manufacturer's root CA certificate, too. But also an operator root CA certificate is imaginable.

Depending on the result of the verification and update process (2) enforced by the AFUP an update response message (3) is signed by the AFUP using its built-in private key PrK_{AFUP}. This also includes the nonce and additional parameters to assure freshness and to support this process by other, optional means (e.g., logging and confirming exact actions that have been taken by the AFUP, reporting of failure events, or even inserting time stamps if these can be provided).

Thus, the OAM server knows the exact state of the AFUP as well of the SW version stored in the network element and can continue with further service actions (which may or may not be transmitted over protected protocols, depending on the security relevance of such action), for instance by initiating a reboot process, as indicated by (4).

Note that such mechanism may involve (and support) additional security and key management processes, which imply, e.g., a secure time base (or at least monotone time counters) for expiry control or for revocation or secure processes to exchange the root CA certificates or private keys, in case this is needed. Also encryption of the SW transfer can be used if confidentiality of the SW is required. Realization and implementation of such issues is a matter of a refined security security specification, which is not further discussed here.

If not done during the reliable boot phase, *initiation of the AFUP* depends on the HW-SW function split and the CPU involvement for message transport, which at run-time may be influenced by SW attacks (potentially causing denial of service).

To prevent such influence the security design might rely on a more sophisticated realization of the AFUP process, in combination with autonomous basic communication capabilities. This would enable *reliable* enforcement of SW updates *at any time*, even in case the network element is compromised by dangerous attacks (e.g., remote SW attacks that however, cannot be directed against the AFUP mechanism if isolated by well designed logic). Accordingly, the AFUP supports remote remediation measures, which cannot be circumvented by such attacks.

B. *Protected CPU / Flash-Memory integration*

The solution presented above needs separate logic for the AFUP mechanism and in its simpler shape (without autonomous communication) it is mainly targeted to assure SW integrity through a (re-)boot process. As an alternative, we can also think of a more flexible realization, where the AFUP is realized by protected firmware being processed by the system CPU.

In the following, we present initial ideas for realization: For security reasons a suited CPU or CPU core is integrated together with an isolated Flash EEPROM, e.g., using Multi-chip modules or dedicated ASICs, as shown in Figure 10. The flash memory might only be accessible in a privileged CPU mode P1 (e.g., controlled by an MMU or by some logic).

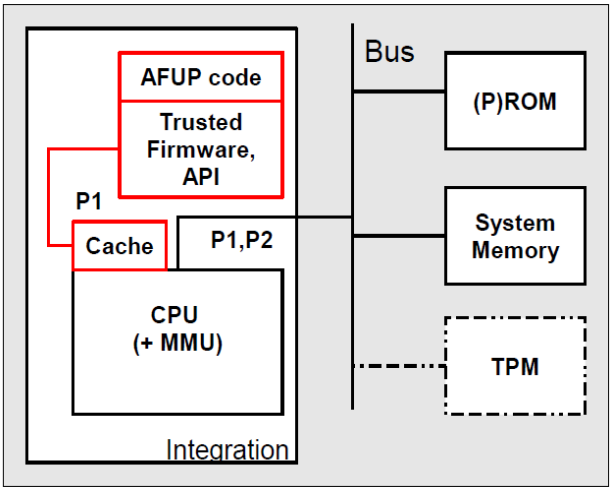


Figure 10. Flexible, protected CPU/Flash memory integration

Trusted functions can only be invoked via a protected API (e.g., by SW-interrupt), assuring that the CPU runs in P1 mode with specific security settings (e.g., indivisible operations, cleared CPU registers etc.). In P1 mode the CPU executes the AFUP process in accordance with the methods and protocols introduced in Figures 8 and 9. Neither external, nor remotely injected SW, nor a local attacker could read or modify any content of the protected flash memory, unless the integrated CPU-Memory device is physically analyzed, requiring extremely high efforts.

In addition to supporting secure boot and the AFUP mechanism this approach could also be used to allow secure run-time integrity protection. To that purpose, trusted API functions could be designed to run checks over parts of the memory content (declared to be invariant), during system operation. Moreover, 'executable' parts of the memory content could be checked and reloaded - periodically or based on events -, in order to wipe out potential hostile modifications that could have been injected during long time operation.

By expanding the above idea on the AFUP functionality for boot-time and run-time checking (of loaded SW) the functionality could also be extended to securely launch any security code (such as a crypto-algorithms) - or even small parts of sensitive general purpose code - at run-time, after successful validation of integrity and authorization. This would be an improvement over the DRTM idea, only allowing for *trusted measurements* on launched code. Such 'authorized SW' could (at run-time) be installed into the trusted memory and externally made available via an extended or updated API.

In addition, such SW could be associated with policies for usage and memory control (e.g., implemented as signed MMU instructions, which could not be changed by 'normal' user-land SW). This would enable a SW security designer to instantiate individual shielded areas of memory, for instance to read- or write protect memory areas being private to a cer-

tain SW module (e.g., to contain derived session keys or even secrets, which could be imported in encrypted form).

VI. PKI AND INFRASTRUCTURE ASPECTS

The presented SWIP concept essentially can be built on (proprietary) Signed Objects SO and on X.509 certificates assigned to signing entities. A X.509 compliant PKI hierarchy might be established, beneficially in manufacturer environment, together with manufacturer specific governance schemes for SWIP. The following observations may substantiate such reasoning:

PKI governance is executed essentially by applying policies associated with the PKI infrastructure, with regard to key and credential management, as well as by organizational control over the involved entities. In accordance with the principles mentioned in Section II.B and E, the conditions for SW signing necessarily must be aligned with the needs for products in mobile network, where each manufacturer individually is responsible for. One impact is the long-term usage (which may be 20 years and more), requiring, e.g., root CA certificates with long expiry periods and related security parameters and capabilities of involved keys. Despite long validity periods there must be an overlapping scheme of valid root CA certificates, which also implies secure exchange of these for products in field for a very long time. Typically this requires issuing of cross certificates as a base for (automated) secure exchange processes, be it via CMP [22] or be it by local means, and sufficient attack protection of the verifying endpoint storing the trust anchor.

Control over the root CA certificates in verifier components is a closely related issue. It must be assured that exclusively such root CA certificates (as well as all intermediate certificates) are accepted, which are compliant with the manufacturer's certificate policies. Such requirements are difficult to fulfill with 'public CAs' (but not impossible, depending on contractual conditions), which typically are designed to meet the requirement of distributed developer scenarios for products with shorter life cycles than those in network environment. Moreover, each product individually may set different conditions for validity (of the SO), for revocation and invalidation, and for SW management and versioning, as well as for the exact mechanisms and rules for verification and enforcement.

In addition to requirements for daily use, it also has to be assured that for exceptional cases (such as 'loss of key material' due to defects or in case of security incidents) disaster and recovery plans are in place and in emergency situations these can be realized very quickly. Even if such incidents (hopefully) are very unlikely to happen, customers may require related features.

Within the manufacturer's development infrastructure protected signing entities have to be established assuring proper usage of associated private keys to sign the SOs for the different products, in accordance with a secure approval work-flow. Such approval work-flow is required to avoid misuse of signing processes for other purposes than those intended by the manufacturer for an individual product. This not only involves personal responsibilities, but also security control such as by appropriate authentication and authorization principles.

Altogether, and in particular with regard to harmonization and generalization (i.e., the different use cases that should be covered) it seems to be the only economic (and perhaps technical) way that manufacturers themselves fully control the environmental conditions and policies for the SWIP infrastructure.

Following such principles the entire SWIP approach is self-contained and may be remotely managed without requiring new specific network infrastructure nodes, neither for modified MTM concepts for secure boot, nor for generalized use cases, such as SW installation or secure SW delivery.

Instead, processes running in TS, OAM or SW management systems might be adapted appropriately. We expect that this could be done in a manufacturer specific way, without the need to standardize commonly agreed solutions.

Note that with the presented approach also protection for SW coming from third parties could be integrated, applying suited extensions for *protection*, *verification*, *enforcement or infrastructure* processes, e.g., by a OEM sided sub-CA, by a manufacturer signed policy that allows a second root , by re-signing SW, or by cross signing of root CAs.

VII. CONCLUSION AND FUTURE WORK

The authors feel that above concepts open a promising way to cover many use cases for SWIP with a harmonized, certificate based approach. It is suited both to cover requirements coming from 3GPP standardization, as well as those that in general increase SW security and reliability for SW products in mobile networks.

One essential benefit is that the same PKI and signing infrastructure could be re-used for many different use cases (e.g., secure boot, SW installation, or integrity monitoring), mainly determined by shaping content, syntax and semantic of SOs and by secure anchoring of adapted verification and enforcement components.

While key points are identified and promising ideas on HW level improvements are tangible (beyond the scope of the native AFUP functionality, as introduced in Section V.B), further research is required, in particular, to solve security issues emerging from cost effective implementation and from long-term operation of network elements.

In practice, trade-offs have to be balanced between achievable security level and efforts for additional HW or CPU modifications, which should be portable among different platforms and CPU types. In our future research work in ASMONIA these issues will be examined, also including virtualization principles. This will go in line with further detailing methods and mechanisms for smooth integration of SWIP management concepts into mobile network elements and security infrastructure.

VIII. ACKNOWLEDGMENTS

Parts of the work presented in this article has been supported by the ASMONIA research project, partially funded by the *German Federal Ministry of Education and Research* (BMBF).

The authors acknowledge the incitations and assistance through the ASMONIA consortium and also like to thank their colleagues at Nokia Siemens Networks for the valuable ideas, discussions, and comments contributing to this work.

REFERENCES

[1] M. Schäfer and W.-D. Moeller, "Strategies for Managed Software Integrity Protection - Managing SW Protection and Trust in Mobile Networks", Proceedings SECURWARE 2010, Fourth International Conference on Emerging Security Information, Systems and Technologies; Venice, Italy, July 2010.

[2] 3GPP TS 33.401, 3GPP System Architecture Evolution (SAE), Security architecture; http://www.3gpp.org/ftp/Specs/html-info/33401.htm, last accessed: January 2011.

[3] 3GPP TS 33.320, Security of Home Node B (HNB) / Home evolved Node B (HeNB); http://www.3gpp.org/ftp/Specs/html-info/33320.htm, last accessed: May 2011.

[4] Trusted Computing Group (TCG), TPM Main Specification, Parts 1-3, Specification Version 1.2, Level 2, Revisions 103, July 2007.

[5] B. Kauer, "OSLO: Improving the security of Trusted Computing"; 16th USENIX security symposium, proceedings, pp. 6-10, August 2007; http://os.inf.tu-dresden.de/papers_ps/kauer07-oslo.pdf, last accessed: May 2011.

[6] R. Sailer, X. Zhang, T.Jaeger, and L. van Doorn, "Design and Implementation of a TCG-based Integrity Measurement Architecture". Proceedings of 13th Usenix Security Symposium, pp. 223-238, San Diego, California, August, 2004.

[7] Trusted Computing Group (TCG), Mobile Reference Architecture, specification version 1.0, revision 1, June 2007.

[8] Trusted Computing Group (TCG), Mobile Trusted Module (MTM) Specification, version 1.0, revision 6, June 2008.

[9] Trusted Network Connect; http://www.trustedcomputinggroup.org/developers/trusted_network_connect, last accessed: May 2011.

[10] W3C Recommendation, "XML Signature Syntax and Processing (Second Edition)", June 2008; http://www.w3.org/TR/xmldsig-core, last accessed: May 2011.

[11] 3GPPP (3rd Generation Partnership Project), http://www.3gpp.org/, ast accessed: May 2011.

[12] J. E. Ekberg and M. Kylänpää, "Mobile Trusted Module (MTM) - an introduction", Nokia Research Center Helsinki, Finland, NRC-TR-2007-015, 2007.

[13] Intel® Trusted Execution Technology (Intel® TXT), Software Development Guide, Measured Launched Environment Developer's Guide, December 2009 (in particular, see Section 1.2.1 therein).

[14] ASMONIA, "Attack analysis and Security concepts for MObile Network infrastructures, supported by collaborative Information exchAnge", BMBF sponsored project; since September 2010, http://www.asmonia.de/, last accessed: May 2011.

[15] ITU-T Recommendation X.509, "Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks", 2008-11.

[16] Online article, "Using Md5 checksums", http://www.openoffice.org/dev_docs/using_md5sums.html, last accessed: May 2011.

[17] Online article, "Maximum RPM: Taking the Red Hat Package Manager to the Limit", Chapter 17. Adding PGP Signatures to a Package, http://www.rpm.org/max-rpm/s1-rpm-pgp-signing-packages.html, last accessed: May 2011.

[18] Entrust Certificate Services, "Java Code Signing",User Guide http://www.entrust.net/ssl-resources/pdf/ECS_Java_Code_Signing_Guide.pdf , November 2010, last accessed: May 2011.

[19] Online Article, "SymbianSigned", http://wiki.forum.nokia.com/index.php/Category:Symbian_Signed, last accessed: May 2011.

[20] K. E. Sanders, SANS Institute, InfoSec Reading Room, Understanding Lotus Notes Security; Execution Control List (ECL) Settings, http://www.sans.org/reading_room/whitepapers/commerical/understanding-lotus-notes-security-execution-control-list-eclsettings_785, last accessed: May 2011.

[21] Cryptographic Message Syntax (CMS), IETF document, Network Working Group, September 2009; http://tools.ietf.org/html/rfc5652, last accessed: May 2011.

[22] Certificate Management Protocol (CMP), IETF document, Network Working Group, September 2005; http://tools.ietf.org/html/rfc4210, last accessed: May 2011.

[23] R. Wichmann, "The Samhain Host Integrity Monitoring System", Samhain User Manual, 2002-2009; http://www.la-samhna.de/samhain/MANUAL-2_3.pdf, last accessed: May 2011 .

[24] A. Apvrille, D. Gordon, S. Hallyn, M. Pourzandi, and V. Roy, "DigSig Novelties", Libre Software Meeting 2005 – Security Topic, slides, July 4-9 2005.

[25] L. Catuogno and I. Visconti, "An Architecture for Kernel-Level Verification of Executables at Run Time", The Computer Journal, Oxford Press, Vol. 47, no. 5, pp. 511-526, September 2004; also: http://www.dia.unisa.it/~luicat/publications/tcj04.pdf, last accessed: May 2011.

[26] T. Schluessler, H. Khosravi, P. Rajagopal, R. Sahita, G. Nagabhushan, and U. Savagaokar, "OS Independent Run-Time System Integrity Services", Corporate Technology Group, Intel Corporation, 2005, see http://www.thefengs.com/wuchang/work/courses/cs592_spring2007/SystemIntegrityServices.pdf, last accessed: May 2011.

[27] ITU-T Recommendation X.500, "Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services ", 2008-11.

Advanced Policies Management for the Support of the Administrative Delegation in Federated Systems

Manuel Gil Pérez, Gabriel López, and Antonio F. Gómez Skarmeta
Departamento de Ingeniería de la Información y las Comunicaciones
University of Murcia, Spain
Email: {mgilperez,gabilm,skarmeta}@um.es

Aljosa Pasic
Atos Origin
Albarracin 25, 28037 Madrid
Email: aljosa.pasic@atosresearch.eu

***Abstract*—Current identity management systems are experiencing an increasing workload of their administrators in the management of the system policies, mainly derived from the sheer amount of policies they have to create and maintain. This problem is even more relevant in federated environments, where roaming users force them to authenticate and authorize people coming from other institutions. In this context, it is increasingly necessary to adopt new advanced policies for the *administrative delegation*, which allow balancing this workload among several delegates who will in turn have a much wider knowledge in the application area where these policies will be applied. In this paper, we present an infrastructure that manages the entire life cycle of the administrative delegation policies in federated environments, as well as a way for reducing the complexity in their management for some scenarios, especially on those where the delegates do not have to be experts in the subject area. These delegates will only have to fill in a simple template, which is automatically generated by our infrastructure. Finally, the applicability of the proposed infrastructure is measured with some performance results.**

***Keywords*-administrative delegation, authorization policies, identity federation, access control.**

I. INTRODUCTION

This paper is an extended and revised version of the conference paper entitled "Advanced Policies for the Administrative Delegation in Federated Environments" [1]. It contains a more comprehensive and detailed explanation of the proposed infrastructure with delegation support, as well as a new section with performance measurements to demonstrate and assess the applicability of the herein introduced prototype in real identity management systems.

Mobility of users among institutions has become more and more common in recent years. For example, the Erasmus Programme [2] has promoted the academic mobility of higher education students and teachers within the European Union. Since the Bologna accords in 1999 [3], and the creation of the European Higher Education Area [4], it is expected this mobility will be increased over time.

On the other hand, we are also currently undergoing the emergence of federated identity systems with the aim of sharing resources among different autonomous institutions. Important examples of these systems are the establishment of academic federations worldwide, such as *eduroam* [5], HAKA [6], or SWITCH [7].

In these scenarios, access control policies are used to manage the access of end users to services and resources offered by an institution. However, as the number of members of an institution increases, new institutions join to the same federation or the relationships among them change, the management of these policies becomes more and more complex. This is due mainly to the great amount of policies to manage, either access control policies, privacy policies or validation policies based on *Levels of Assurance* (LoA) [8], among others.

To reduce the complexity in the management of these policies the *administrative delegation* allows system administrators to delegate some privileges to others, named *delegates*, with the aim of making part of their work by managing a subset of the system policies [9]. In this way, not only is the management of the system policies distributed to other people, but also they are being delegated to people who have better knowledge on the application area upon which these policies will be used.

As an example, the system administrator of an institution may delegate in the head of a department to specify which of the members of her department can access the network. This delegate will be also able to establish certain constraints under which her employees can do it, e.g., they will be only able to access the network in an specific time interval.

This new sort of policies supposes a new value-added service to the current policy-driven systems, either federated or not, although its use also introduces some drawbacks that have to be treated adequately:

- The number of policies to manage increases dramatically. System administrators will have to manage both the policies that already existed (access control policies or privacy policies, among others) and this new kind of policies to control the administrative delegation. It will introduce a new way of controlling which users can create new policies (administrative policies).
- Delegates are usually users with no knowledge in policy management, access control languages, etc. Thus, we should make it easier for those people the generation

and management of this new kind of policies.

As seen, although the workload of administrators is reduced, or distributed considerably among several delegates, the policy management (including the administrative ones) will also be more complex. Thus, the definition of these new policies for the administrative delegation is not enough for its deployment in real environments, but it is also necessary to define an infrastructure that can manage them and help delegates to do their new tasks.

As a solution to these problems, we will include a set of new components to existing federated identity systems to manage the complete life cycle of the administrative delegation policies. Moreover, we will also define a new mechanism for the generation of templates that helps delegates to carry out this new task in a simple and intuitive fashion. These templates will be automatically generated by our infrastructure from the administrative delegation policies created by the system administrators.

The remainder of this paper is organized as follows. Section II describes an example scenario that is used to motivate this research work. The access control language used in this proposal and its extension to enable the use of the administrative delegation are shown in Section III. Section IV describes in detail the infrastructure providing administrative delegation, whereas Section V presents the automatic generation of policies and templates for helping delegates to do their tasks. Section VI illustrates some performance measurements to assess the applicability of the prototype in real identity management systems. Then, Section VII presents the main related work and, finally, Section VIII remarks the main conclusions and future work.

II. USE CASE

As an application example of this new kind of policies let us suppose a scenario where an institution is going to host an international project meeting, in which members from other institutions need Internet access. Then, the host institution will provide such a connection with certain *Quality of Service* (QoS) assurances. All participants, coming from various institutions, belong to the same identity federation.

Figure 1 depicts this situation, where two participants coming from different institutions (Bob from *Institution A* and Carl from *Institution B*), but all belonging to the same identity federation than the host institution, want to get a connection to the Internet.

The administrative delegation can be used in this scenario to assign the responsibility of managing the access control properties to a user more closely related to the mentioned scenario. For instance, the person who is organizing the meeting; she knows all the necessary information, such as the identity of the audience or the meeting schedule. In this way, the host institution's administrator will delegate to the meeting organizer, i.e., the delegate, the definition on which participants will have access to the network, as well as the schedule upon which they can do it; information the meeting organizer perfectly knows.

For this example, let us suppose Alice is the meeting organizer, or delegate, who will define all access control policies for the meeting participants. Then, the system administrator, besides establishing this delegation of privileges to Alice, will only have to define the QoS assurances the system should apply. This information are network parameters too much technical the delegate does not need to know.

We can prove with this scenario the use of the administrative delegation, where it prevents the administrator has to create access control policies on a group of people he does neither know nor has information with respect to the requirements each one needs.

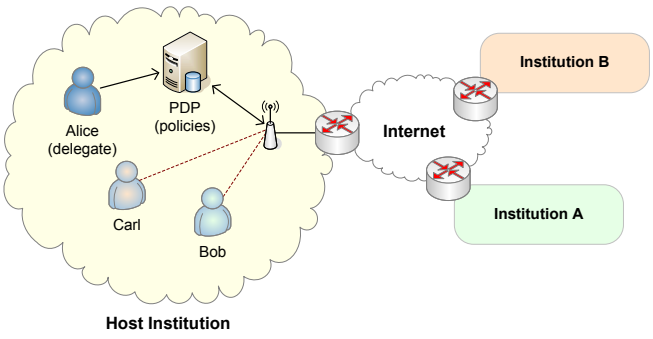


Figure 1. Example of an international project meeting

Another interesting example related to the administrative delegation can be found in multi-stakeholder scenarios such as outsourcing [10]. Some business processes are relying on IT systems of contracted service providers, the complexity of evidence collection is very high. The business processes that are subject to compliance are often scattered across multiple business units in a variety of unorganized and unmanaged systems, so design and implementation of internal control processes are not an easy task.

In this context, the MASTER EU-IST project [11] focuses, among other research topics, on automation of evidence collection. This is done with the support of MASTER operational infrastructure that relies on a set of indicators, measurable and observable properties derived from system events and configuration policies, which are set up at various levels in order to ensure trustworthy control process. What makes MASTER especially interesting is the possibility to separate evidence collection from posterior evidence correlation or compliance assessment.

In the emerging service delivery models, e.g., cloud computing, there is often an outsourcing chain where customers contract a service provider in order to do *Business Process Outsourcing* (BPO). In its turn, these service providers can store data in cloud or use some other cloud computing resources, which is not necessarily belonging to the same identity federation. As a consequence, the service provider

may not be able to offer all the required evidences to the customer. Thus, the customer is constrained to the offered granularity and semantics of the provided events, as well as monitoring and enforcement capabilities of the service provider.

Another constraint for the administration of evidence-collection process in multi-outsourcing context is the perceived lack of trust. Customers might believe that events or related evidences provided by service providers (or their subcontractors) are not authentic.

The administrative delegation mechanism proposed in this paper offers novel ways to increase the trust level. On the one hand, the trust level could be increased by applying more distributed monitoring and configuration policy rules, as well as fine grained access control policy to MASTER components and policies. Increased number of policies and decoupling of components responsible for signaling and monitoring of events (both needed for evidence collection) increases also complexity and administrative risks, which in its turn can be also addressed with administrative delegation where internal control process owner may delegate or specify who can have access to which evidence collection components. This way, each configuration of the MASTER operational infrastructure corresponds to a set of component access control policies, managed by the person who will have more knowledge about her employees than the client or service provider control process owner.

III. Administrative delegation in XACML

As mentioned before, we have identified the administrative delegation as a good alternative to manage the policies of complex systems. This section defines the mechanisms included in the eXtensible Access Control Markup Language (XACML) [12], a standard XML-based access control language, to allow the use of this new feature.

This proposal was defined to represent access control policies in a standard way. It includes two different specifications: the first one is an access control policy language, which defines the set of *subjects* that can perform particular *actions* on a subset of *resources*; the second one is a representation format to encode access control requests (a way to express queries about whether a particular access should be granted) and their related responses.

The administrative delegation in XACML relies on the idea that a person authorized to delegate certain privileges does not need to use them, and vice versa; that is, that a person owns rights to exercise a privilege does not imply that she can delegate it to others.

As the XACML delegation profile [13] specifies, the purpose of the delegation model is to make it possible to express permissions about the right to issue policies and to verify the issued policies against these permissions. This profile defines a new XML element, named *PolicyIssuer*, to indicate who has issued a specific policy. Through this element, the system can identify and verify whether the policy issuer is valid to delegate the given privilege before being applied. Thus, the authority of the issuer needs to be verified in order to consider this policy as valid. When a policy does not include the *PolicyIssuer* element, the policy is considered trusted and, therefore, it is valid.

By including this new element, the administrative delegation allows the creation of *delegation chains*, where a certain privilege can be delegated from one person to another. For example, in the use case shown in Section II, the meeting organizer (Alice) could in turn delegate in Bob the generation of the access control policy for Carl, thus building a delegation chain of three policies: an administrative policy, created by the administrator, that delegates in Alice the privilege of accessing to the network; another administrative policy, created by Alice, that delegates in Bob the same action and resource; and the access control policy, created by Bob, that finally grants access to the network to Carl.

In a schematic way, this delegation chain would be as follows (each arrow represents a delegation policy):

$$Administrator \rightarrow Alice \rightarrow Bob \rightarrow Carl$$

Each delegation policy can specify another XML attribute, named *MaxDelegationDepth*, to limit the depth of delegation that is authorized by such a policy. In this example, if the system administrator decides to restrict the delegation chain to only one person, by defining *MaxDelegationDepth="1"*, Alice will not be able to delegate her privileges to anyone else. Otherwise, the delegation chain will not be trusted and Carl will not have access to the network.

Thus, besides the traditional policies for managing the access to the resources (*access policies*), and the requests the *Policy Decision Point* (PDP) receives, which should be resolved based on these policies (*access requests*), this new specification also defines a new set of XACML policies to validate the issuer of another policy (*administrative policies*). This set of policies will be consulted to the PDP by means of another sort of requests (*administrative requests*).

As an example, Figure 2 depicts the delegation model in XACML for the use case presented in Section II. As seen in this figure, once the PDP receives an access request, and the policy used for its evaluation includes the *PolicyIssuer* element, as the one shown in Figure 2a, it is necessary to carry out an administrative request to verify whether the policy issuer is trustworthy, and she has the expected permissions for such a delegation. This administrative request is built from the previous access request, which will include the attributes of both users (Alice and Bob) gathered from their home institutions.

Finally, the administrative request is evaluated using the corresponding administrative policy, as the one shown in Figure 2b. It is worth noting that if this administrative policy in turn contains the *PolicyIssuer* element, the above process must be repeated until either reaching a trusted policy or

exceeding the maximum delegation depth defined in the delegation chain.

In this scenario, the policy issuer (whose identifier is *Alice*) authorizes, through the access policy, to another person with identifier *Bob* (subject) to *Access* (action) the *Network* (resource). It is also included a time condition indicating the period of time within which the policy is valid, i.e., the meeting schedule. The previous access policy is conditioned to the issuer is recognized as valid for carrying out such a delegation. To this end, the administrative policy of Figure 2b permits that the access to the network can be managed by those delegates who have the attribute *schacUserStatus* with the *meeting:set* value, and the additional condition that users requesting the access own the attribute *schacPersonalPosition* with the *Researcher* value [14]. Thus, if Alice and Bob comply with these attributes, both policies will be evaluated as valid and, finally, Bob will have the requested network connection. In this case, an obligation is also attached, a QoS requirement, which has to be enforced in the system by the *Policy Enforcement Point* (PEP).

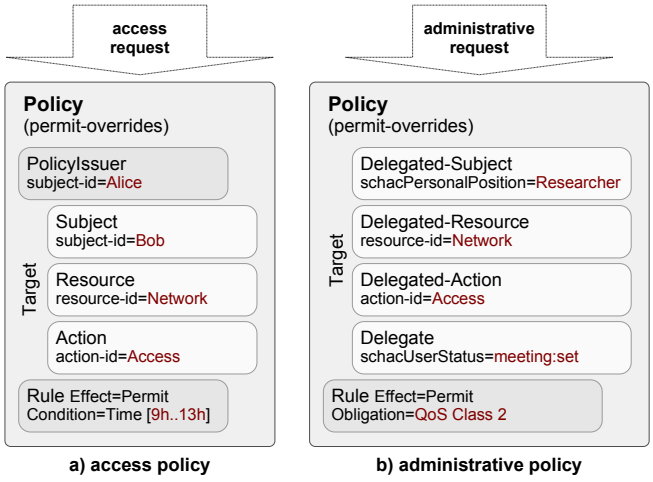


Figure 2. Administrative delegation in XACML

As has been seen in this section, the XACML delegation profile defines the syntax for the new elements that provide administrative delegation support. However, it is necessary an infrastructure that makes easy the use of this new sort of policies, as well as some mechanisms to help, especially to the delegates, in the management of these policies in an easy and intuitive fashion. Both of them are described in detail in the following sections.

Furthermore, XACML is only focused on intra-domain systems without providing any support to federated environments. Therefore, we must also take into account the identity management in our administrative delegation architecture for inter-domain systems. In this case, the system administrator will be also able to delegate part of her duties not only to people of his very institution, but also he will be able to do it to outsiders as long as they belong to the same federation his institution.

IV. Infrastructure with delegation support

Once the application area and the language to express delegation have been presented, this section describes the proposed infrastructure that makes use of delegation policies.

A. System requirements

The needs and the minimum requirements that any administrative delegation system should comply with are summarized as follows:

1) The institution wants to offer an administrative delegation service is required to provide a secure repository where to store the administrative policies.
2) The administrative policies have to be published and stored in an internal repository through secure channels that provide features of confidentiality, authentication and integrity. Moreover, it is also advisable that these secure channels provide *non-repudiation* features. This last property will avoid delegates can refuse the creation and modification of those policies for which they are responsible.
3) Efficient authentication methods are required. Only authorized people can both access the secure repository and exclusively create or modify the policies to which they have permissions. In this case, administrators will have a total access and control to all stored policies, while delegates will be only able to create and modify those policies to which the administrator has provided access, creation and modification rights.
4) The secure repository with the administrative policies has to be accessible by the service that takes the authorization decisions, i.e., the PDP, according to such policies.
5) The delegate should be capable of defining policies without having technical knowledge in managing access control policies.

B. Federation environment

An identity federation system is composed by several institutions in which a set of common services are offered, such as the authentication and authorization of the federation members. In this way, when a roaming user moves from her *home institution* to another, the *visited institution*, the later can authenticate her through the federation. In some scenarios, this authentication phase is carried out remotely by the user's home institution.

For example, *eduroam* is one of the largest networks for roaming worldwide, oriented to institutions involved in research and education [5][15]. This network allows the mobility of users across over 40 countries throughout three continents, including China, Australia and Canada.

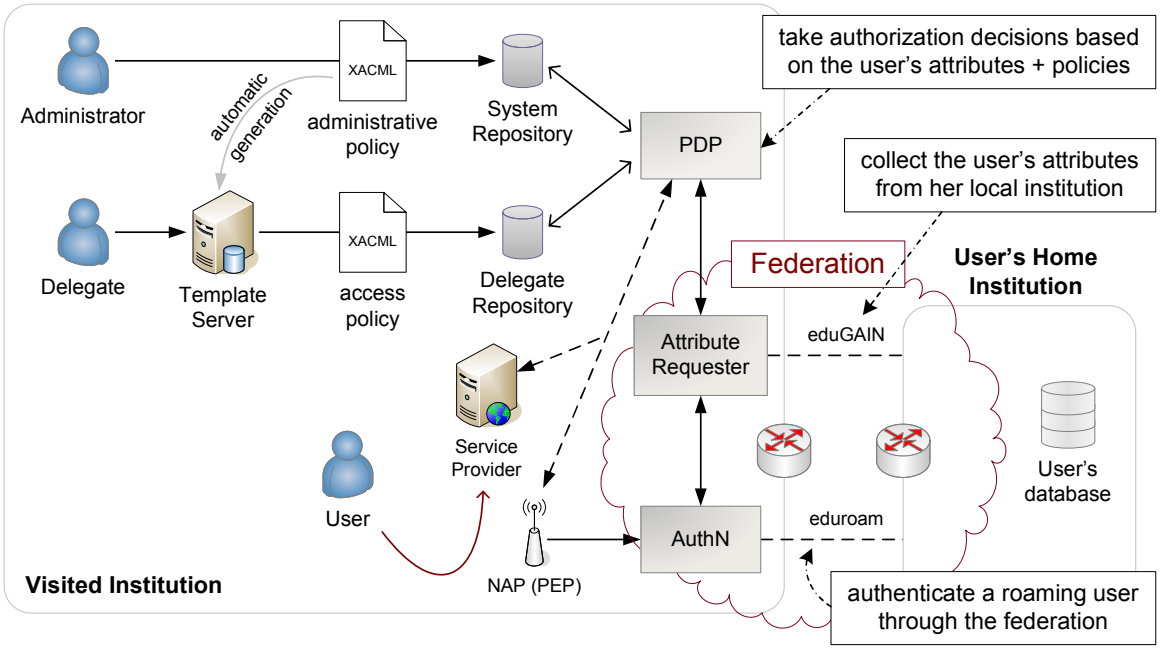


Figure 3. Administrative delegation architecture

The *eduroam* network is based on a RADIUS hierarchy that redirects the user's authentication request to the appropriate home RADIUS server. During this phase, the user receives some kind of handler to identify her in the federation, thereby providing a way of preserving her privacy in the federation. Later on, if the visited institution needs some user's attributes, to take a local authorization decision by the PDP, an attribute request can be made through the federation using the previous handler.

To that end, each institution of the federation must host two different entities (see Figure 3): an *AuthN* module to authenticate a remote user through the federation, returning her a handler (opaque identifier); and an *Attribute Requester* module that returns the attributes associated to the user identified by the previous handler. In our case, the authentication is based on *eduroam*, whereas the attributes request is carried out through *eduGAIN* [16][17]. Finally, authorization decisions are locally taken in the visited institution, according to these user's attributes and the system policies defined by the institution.

This identity federation system is a good testing environment to make use of delegation policies in a feasible way.

C. Components of the architecture

The two services previously defined forward the corresponding requests to the user's home institution. Apart from them, the proposed architecture (shown in Figure 3) also introduces a PEP in order to protect the resources, enabling the access to the authorized users, and a PDP to take the authorization decisions based on the defined XACML policies. The PDP also needs the user's attributes to evaluate these policies, which are harvested through the Attribute Requester for remote users and directly from the *User's database* for local users.

It is worth noting that the protected resource in the example shown in Figure 3 is the network, so that the PEP is the *Network Access Point* (NAP) that provides both wireless connectivity and access control capabilities according to the PDP's decisions.

Moreover, in this architecture we include two different repositories to store all the institution policies:

- *System Repository*. This contains both the traditional policies of the institution, e.g., access control policies or privacy policies, and the new administrative policies. All of them are created and managed by the system administrators.
- *Delegate Repository*. This repository contains all the access control policies generated by the delegates.

The reason of maintaining two different repositories is mainly the prevention of some security issues. The first repository, with the system policies, contains critical policies for the correct operation of the institution. This means that granting the access to other people may cause security risks.

Once the *Delegate* has been enabled as the person in charge of defining some system policies on behalf of the administrator, this user can access the *Template Server* to do this task. The Template Server is a Web application server that allows delegates to define access control policies from the privileges the system administrator has delegated on them. This component stores the templates, in the

form of Web-based forms, that delegates can fill out in a straightforward manner without having technical skills about policies management.

To do so, the Template Server maintains a third repository, apart from the two ones introduced above, where the templates are stored. These templates are automatically generated from the administrative policies created by the system administrators. Thus, delegates will then access to this server when they need to create an access control policy, according to the delegation model presented in Section III.

The Template Server is not very different to any other service that the federation offers, since this server will also make use of the defined mechanisms for controlling the access control to the templates. This decision is based on the credentials that the delegate owns in the institution to which belongs.

Therefore, delegates will have to authenticate in their home institution and, if both institutions belong to the same federation and delegates have the appropriate credentials, they will be able to access the Template Server.

The new components we have added in the identity federation system, shown at the top of Figure 3, can operate both in federated environments (inter-domain) and in autonomous institutions (intra-domain). The only difference between them lies on where and how the users' authentication and authorization are taken place.

In an autonomous mode of operation, all these processes are performed in the institution locally, whereas in a federated environment, as the one shown in Figure 3, the authentication is performed remotely at the user's home institution through *eduroam* and the authorization is carried out in the visited institution in a local way.

In the latter case, the authorization phase needs the user's attributes for the decision-making process. The visited institution should then interact with the user's home institution (note that both of them have to belong to the same federation) for recovering these attributes through *eduGAIN*.

D. Delegation policies

In order to take advantage of the administrative delegation, which is defined in the XACML guidelines detailed in [13], we have to define two different kinds of policies. On the one hand, an administrative policy expressing the delegation of the administrative rights to the delegates and, on the other hand, the access policies created by the delegates containing the details of the access control rules.

Using the use case presented in Section II, we have defined two example policies. The first one, the administrative policy, shown in Listing 1, specifies that a user who holds the *schacUserStatus* attribute with the *meeting:set* value can act as a delegate, and grant *Access* to the *Network* to any user who holds the *schacPersonalPosition* attribute with the *Researcher* value.

This policy also specifies that the maximum delegation depth is set to 1, by means of the *MaxDelegationDepth* attribute. Moreover, this administrative policy specifies some network properties to enforce; in this case, a QoS assurance with the *Class 2* value.

```
<Policy PolicyId="AdministrativePolicy" RuleCombiningAlgId="permit–overrides"
      MaxDelegationDepth="1">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string–equal">
          <AttributeValue DataType="string">Researcher</AttributeValue>
          <AttributeDesignator AttributeId="schacPersonalPosition"
                Category="...:delegated:...:subject–category:access–subject"/>
        </Match>
      </AllOf>
    </AnyOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string–equal">
          <AttributeValue DataType="string">Network</AttributeValue>
          <AttributeDesignator AttributeId="...:resource–id"
                Category="...:delegated:...:attribute–category:resource"/>
        </Match>
      </AllOf>
    </AnyOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string–equal">
          <AttributeValue DataType="string">Access</AttributeValue>
          <AttributeDesignator AttributeId="...:action–id"
                Category="...:delegated:...:attribute–category:action"/>
        </Match>
      </AllOf>
    </AnyOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string–equal">
          <AttributeValue DataType="string">meeting:set</AttributeValue>
          <AttributeDesignator AttributeId="schacUserStatus"
                Category="...:delegate"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Rule RuleId="AdministrativeRulePermit" Effect="Permit">
    <Obligations>
      <Obligation FulfillOn="Permit">
        <AttributeAssignment AttributeId="QoS">
          Class 2
        </AttributeAssignment>
      </Obligation>
    </Obligations>
  </Rule>
</Policy>
```

Listing 1. Administrative policy

On the other hand, the access control policy example is shown in Listing 2. This policy is issued by *Alice*, who grants *Access* to the *Network* to *Bob*. This policy could also include some conditions and network parameters that will have to be enforced by the PEP. In this case, Alice establishes that this access is only permitted from 9h to 13h.

```
<Policy PolicyId="AccessPolicy" RuleCombiningAlgId="permit–overrides">
  <PolicyIssuer>
    <Attribute AttributeId="...:subject–id">
      <AttributeValue DataType="string">Alice</AttributeValue>
    </Attribute>
  </PolicyIssuer>
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string–equal">
```

```
            <AttributeValue DataType="string">Bob</AttributeValue>
            <AttributeDesignator AttributeId="...:subject–id"
                  Category="...:subject–category:access–subject"/>
          </Match>
        </AllOf>
      </AnyOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="string–equal">
            <AttributeValue DataType="string">Network</AttributeValue>
            <AttributeDesignator AttributeId="...:resource–id"
                  Category="...:attribute–category:resource"/>
          </Match>
        </AllOf>
      </AnyOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="string–equal">
            <AttributeValue DataType="string">Access</AttributeValue>
            <AttributeDesignator AttributeId="...:action–id"
                  Category="...:attribute–category:action"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
    <Rule RuleId="AccessRulePermit" Effect="Permit">
      <Target/>
      <Condition>
        <Apply FunctionId="function:and">
          <Apply FunctionId="function:time–greater–than–or–equal">
            <EnvironmentAttributeDesignator AttributeId="current–time"/>
            <AttributeValue>Wk0900</AttributeValue>
          </Apply>
          <Apply FunctionId="function:time–less–than–or–equal">
            <EnvironmentAttributeDesignator AttributeId="current–time"/>
            <AttributeValue>Wk1300</AttributeValue>
          </Apply>
        </Apply>
      </Condition>
    </Rule>
</Policy>
```

Listing 2. Access control policy

E. Conditions and obligations

As seen before, both policies define two sets of conditions and obligations (one per policy), which should be managed by the PDP. Thus, these conditions have to be combined each other and check them later to know they do not enter in conflict, or they are not contradictory. For example, the administrative policy could indicate that the user can only access the network from 8h to 14h, whereas the access control policy created by a delegate can restrict this period of time from 9h to 13h. In the policy evaluation process, which is carried out by the PDP, the conditions of the access control policies will be checked first, and the ones of the delegation policies later. As a result, the intersection of both constraints will be enforced, which is a correct way of operation.

On the other hand, the network parameters can produce some kind of inconsistency if the ones included in the access control policy are inconsistent with the ones established in the administrative policy. Although the correct way of managing the different kinds of network parameters depends on their type, in general, the values derived from the access policies can only be a subset of the ones derived from the administrative policy.

In this sense, delegates cannot grant wider privileges than the ones specified by the administrator in the administrative policy. In case of conflict between the properties stated by both policies, the PDP must ensure that the properties provided by the access control policy will be enforced iff they are less or equal than the ones permitted by the administrative policy.

V. Automatic Generation of Policies and Templates

The first step in the system is to define the administrative policy. This is done by the system administrator in a usual way, although in order to make this task easier the administrator could use any kind of XACML policy editor.

Among the available policy editors, all of them developed under an open source license, we can stand out:

- XACML-Studio [18]. This is an authorization policy editor implemented as a Web application to import, create, edit and export polices in XACML 2.0 format.
- XACML.NET [19]. This editor is completely developed in .NET/C#, which implements almost the entire XACML 1.0 standard, excepting the *regexp-string-match* function.
- eXist [20]. eXist-DB is a database capable of storing and managing information in XML format in a native way, as well as processing XQuery and XPath queries on the database itself. The policy editor is embedded within the editor itself that eXist provides to manage all the policies stored in its database. Anyway, this supposes a dependency drawback since it forces our infrastructure to use such a database to manage the policies from this editor. Moreover, it only supports the versions 1.1 and 1.0 of the XACML format.
- UMU-XACML-Editor [21]. This editor is implemented in Java with the aim of creating policies by following the XACML 2.0 standard.

Although UMU-XACML-Editor, as previous ones, does not support the latest XACML specification (version 3.0) we have chosen it as the policy editor for this research work.

This solution is the policy editor most updated of the existing ones, in addition to some interesting technical features such as defining references to other policies and supporting the *SchemaLocation* element to validate polices against their XML schemes. Thus, the UMU-XACML-Editor policy editor has been extended to support the new XACML delegation profile, as presented in Section III.

Once the administrator has defined the administrative policy by using the UMU-XACML-Editor, the next step is the definition of the access control policies by the delegate. But as indicated previously, how the delegate is going to create the access control policies is a tricky aspect, because she is a common user and not a security expert with skills on policy languages.

Therefore, instead of building this policy from scratch, some kind of template (in our case, Web-based forms) can be provided to the delegate to help her in this administrative

task. In this way, the delegate will only have to fill in this template to create the appropriate access control policy.

A. Generation of templates

To perform this process is necessary to use a *Policy Management Tool* (PMT) with the aim of building the access policy templates. In our case, we have decided to extend the UMU-XACML-Editor policy editor to include this new feature. This tool will use an XSLT [22] transformation that extracts the appropriate fields from the delegation policy to generate the template.

The PMT needs to search for the *action* and *resource* elements specified in the administrative policy, which are identified by the "*...:delegated:...:attribute-category:action*" and the "*...:delegated:...:attribute-category:resource*" categories (see Listing 1). The rest of data needed for the access control policy, such as the subject identifier, are included in the template as input fields for being filled in by the delegate. The identifier of this delegate is automatically added in the new policy as the issuer thereof.

This process is depicted in Figure 4. Note that the text enclosed in parentheses indicates which element of the infrastructure (the administrator, the delegate or the system itself) generates each piece of information.

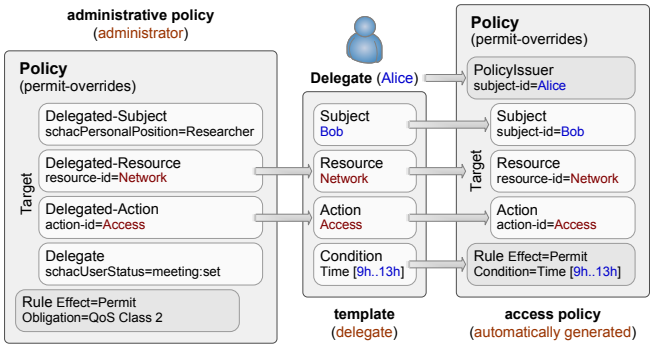


Figure 4. Administrative delegation process

Besides, the template must allow the delegate to specify the conditions and obligations. The most common condition is the time constraints although, however, other conditions such as a maximum number of connected users could be included in it. Regarding the network parameters, the template can also show to the delegate the different parameters defined in the system that she can assign, e.g., QoS or bandwidth, together with their possible values.

Once the PMT has generated this template, it is necessary to make it available to the delegates on some server or repository. But the access to this template must be restricted to the appropriate delegates, so that the PMT should also generate an XACML policy to control the access to it. In this case, the PMT must search for the "*...:delegate*" category in the administrative policy, which specifies the restrictions for the delegates. This field is then used to specify the subject of the policy to control the access to the template. Finally, both the administrative policy and the access control policy to the template are stored in the System Repository to be used later.

As we can see in Figure 5, from the administrative policy created by the administrator, the PMT is also capable of: (1) generating the template for the delegate; and (2) generating the corresponding policy to control the access to such a template. Both policies are stored in the System Repository with the system policies, whereas the template is stored in the internal repository of the Template Server.

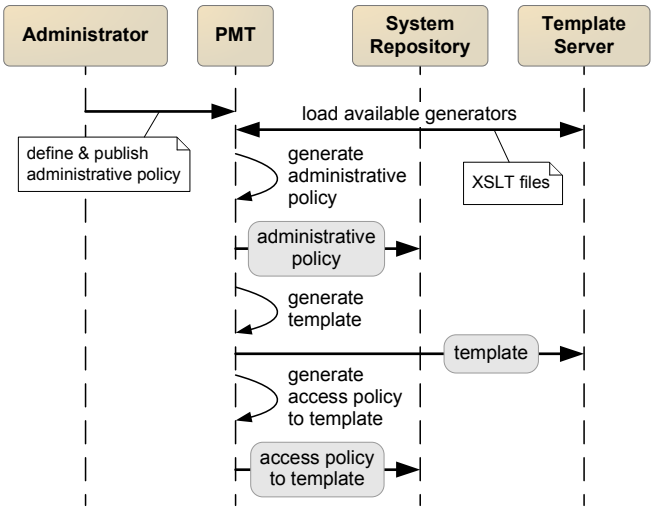


Figure 5. Generation of automatic templates

In order to control the access to these templates, the Template Server has to query the PDP to take an authorization decision according to the *access control policies to templates* previously generated by the PMT.

B. Generation of access control policies

Figure 6 shows the complete process for the generation of access policies from the previously generated templates. Before allowing delegates to access to these templates, the Template Server should determine whether the delegate owns the required credentials and privileges to take advantage of an administrative delegation.

To do so, the delegate must first be authenticated through the AuthN module (see Figure 3). At this point, it is worth pointing out that this user will be authenticated in her home institution, either locally if the delegate belongs to the same institution or remotely through *eduroam* if she belongs to a remote institution with which the visited institution has a close relationship through the same identity federation.

If the authentication has been successful, the Template Server has to collect the delegate's attributes to decide if this user owns the minimum privileges to complete this administrative task. This harvest process is performed through the Attribute Requester module (see Figure 3), which will

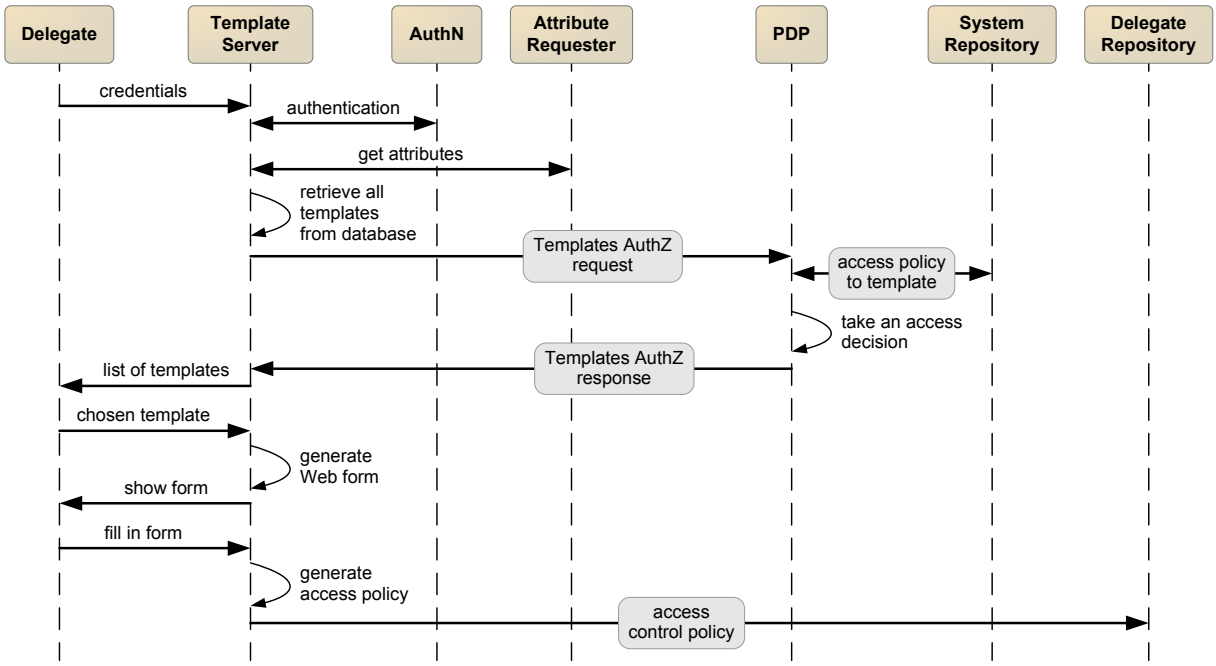


Figure 6. Process of using templates by the delegates

gather them from the home institution to which the delegate belong. As before, this process will be done either locally or remotely (in this case, through *eduGAIN*), depending on which the delegate's institution is.

After this harvesting process, the Template Server retrieves all the defined templates from its internal repository. Then, the Template Server builds an authorization decision request that sends to the PDP to know which of these templates the delegate can fill in.

```
<Request>
  <AuthorizationDecisionQuery Resource="Template-11">
    <Subject>
      <NameIdentifier NameQualifier="https://idp.um.es/shibboleth/idp">
        Alice
      </NameIdentifier>
    </Subject>
    <Action Namespace="urn:segura:umu:actions">Fill-in</Action>
    <Evidence>
      <Assertion>
        <AttributeStatement>
          <Attribute AttributeName="eduPersonPrincipalName"
                AttributeNamespace="urn:segura:edugain:attributes">
            <AttributeValue>alice@um.es</AttributeValue>
          </Attribute>
          <Attribute AttributeName="schacUserStatus"
                AttributeNamespace="urn:segura:edugain:attributes">
            <AttributeValue>meeting:set</AttributeValue>
          </Attribute>
        </AttributeStatement>
      </Assertion>
    </Evidence>
  </AuthorizationDecisionQuery>
</Request>
```

Listing 3. Access request to verify if the delegate can fill in a template

This authorization decision request must include:

- the attributes of the delegate in the *Assertion* element;
- the template to be accessed in the *Resource* element;
- and the *Action* element with the *Fill-in* value.

An example for the use case presented in Section II can be found in Listing 3. In this case, the PDP is evaluating whether Alice can fill in the template number 11 or not. If so, the PDP will return to the Template Server a signed authorization decision response stating that *Decision="Permit"* to the *Resource="Template-11"* for the *Action="Fill-in"*.

During this decision-making process, the PDP makes use of the access control policies to templates, which have been previously stored in the System Repository by the PMT.

Finally, the Template Server will show to the delegate the list of templates that can fill in according to the presented credentials. A screenshot of this Web page can be found in Figure 7. At the top of this figure is shown the graphical result obtained by Alice after entering her credentials in a previous phase. In this case, Alice can only fill in the template number 11 with the "*Meeting Segur@ Internet Connection*" description. At the bottom of the same figure, we can see the logging server output to check the different steps explained in this section.

Once the delegate selects one of the available templates, the Template Server internally generates the Web form that will return her so it can be finally filled in. Continuing the previous example, Figure 8 depicts another screenshot where Alice has already filled in the Web form with the requested information. As can be seen, the administrative definition of a new access control policy is a very straightforward process that does not require special skills in managing policies.

Finally, and after filling in the template, the Template Server internally generates the access control policy and

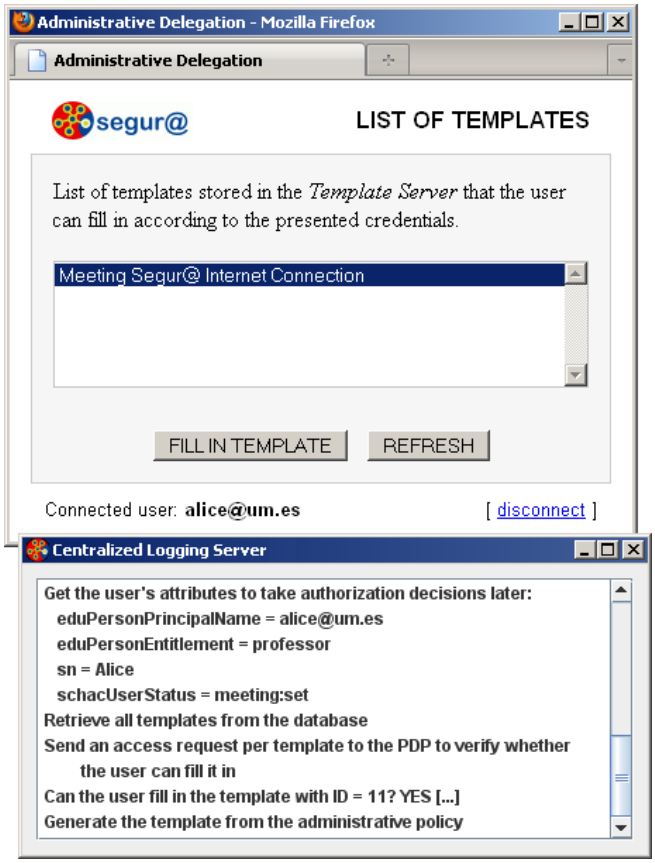


Figure 7. List of templates the delegate can fill in

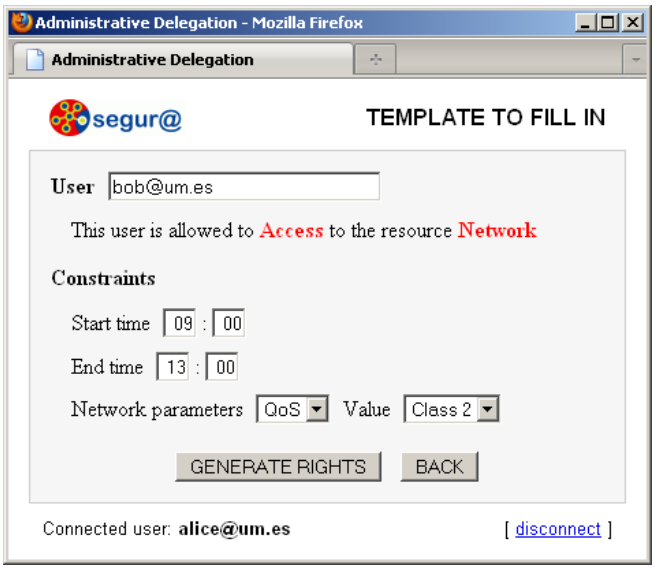


Figure 8. Template filled in by the delegate

stores it in the Delegate Repository for uses in future requests from *Bob* to *Access* to the *Network*, provided this request is from 9h to 13h. For this generation we also make use of XSLT transformations, as seen in the right-hand side of Figure 4.

VI. Performance Results

The infrastructure with delegation support proposed in Section IV has been implemented and deployed in a lab testbed to demonstrate its applicability in a real scenario.

In these tests, we have assumed that both the administrator and the delegate belong to the same institution, so no authentication process is carried out remotely through *eduroam*. As a consequence, the remote harvesting process for getting the delegate's attribute is not performed through *eduGAIN*. In its stead, all these processes are taken in the same institution.

This validation has been performed through some performance measurements, which have been taken directly from this testbed, depending on different factors that might have an important impact on the proposed administrative delegation process. These tests are:

- Retrieve the list of templates from the Template Server that a delegate can fill in. In this process, we also include the authentication and authorization phases of the delegate when she accesses to the Template Server for first time.
- Validation of an access control policy sent to the PDP by a user.

In both cases, we have assumed for these tests our infrastructure is configured with the policies and features presented throughout this paper. The list of hardware and software requirements deployed in our lab testbed is shown in Table I.

A. Retrieve the list of templates from the Template Server

This first test aims to assess the time a delegate needs to access the Template Server in order to fill in some of the templates for which she is responsible. It will give us an idea about the time consuming on the different phases involved in this process, as detailed in Section V-B, which have been divided in four different steps (all of them corresponding to the arrows depicted in Figure 6):

- *User AuthN*. Time needed by the infrastructure to authenticate the delegate. This step corresponds to the *authentication* row depicted in Figure 6.
- *User AuthZ*. This step represents the time needed to retrieve the delegate's attributes by means of the Attribute Requester. This corresponds to the *get attributes* arrow.
- *DB Templates*. It corresponds to the *retrieve all templates from database* arrow. This retrieval is internally done from the internal repository of the Template Server.
- *Templates AuthZ*. This step pertains to the decision-making process that the PDP has to do to know which templates the delegate can fill in.

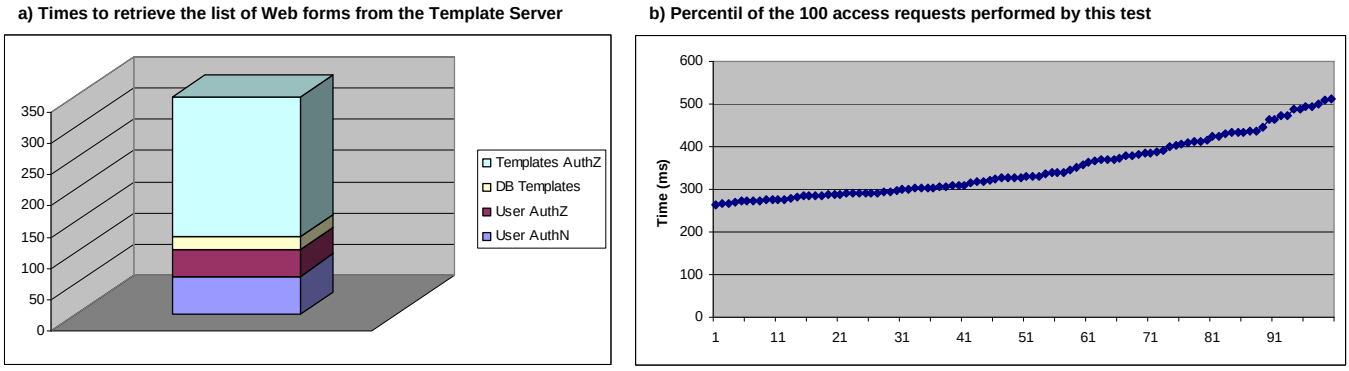


Figure 9. Administrative delegation process

	Hardware	Software
Template Server	Intel Pentium 4 CPU 640 CPU: 3.20 GHz, 32 bits Cache size: 2048 KB L2 Total memory: 1024 MB	Windows XP SP3 Tomcat 6.0.10
PDP	AMD Opteron CPU 246 CPU: 2.0 GHz, 64 bits Cache size: 1024 KB L2 Total memory: 1024 MB	Ubuntu 9.10 Tomcat 5.5.27 eXist-DB 1.2.6
AuthN & IdP		Ubuntu 9.10 Apache 2.2/mod_ssl MySQL 5.0.75 OpenLDAP 2.3.28
Attribute Requester		Ubuntu 9.10 Tomcat 5.5.27

Table I
HARDWARE AND SOFTWARE REQUIREMENTS

The testing process has been performed by means of sending 100 sequential requests, from which we have extracted the average times of each these four steps. Note that all times have been measured in milliseconds (ms). The partial times of each of them are shown in Figure 9a, while Figure 9b illustrates the total times for the 100 requests.

We can observe in Figure 9a what is the time distribution for each case. At first sight, we can assert that the decision-making process performed by the PDP is the longer time of all, as expected, being the 63,81% of the total time. The *Templates AuthZ* step takes 224 ms, on average, for the 100 tests performed, while the total time takes 351 ms. For this process, the PDP only needs to evaluate the access control policies to templates, so no delegation chain is used by it.

It is worth noting that these times have been taken from the Template Server. This means that the PDP does not really take these 224 ms, because we have to take in consideration the network traffic and delays. We have measured this time, and the PDP really takes 198 ms on average in this process of making a decision. With this last time, it would be now a 54% of the total time. Apart from that, this time is very reasonable since the PDP has to take its decision, build the corresponding SAML response and sign it digitally.

In any case, as a conclusion, these times are perfectly acceptable and assumed by the delegate.

B. Validation of access control policies

As have seen in the previous test, the PDP is the critical component that can slow down the performance of the overall system. Then, for this second test we have taken into consideration how the delegation chain defined in Section III can influence in this performance.

That delegation chain stated that the administrator delegated to Alice the definition of the access control policies for such a scenario. Thus, once Bob tries to access the network the PDP should check both the administrative policy, created by the administrator, and the access control policy created by Alice. In this process, the PDP will also have to retrieve the Alice's attributes for making the corresponding decision. Note that the retrieval of these attributes will be done through the Attribute Requester.

For this test, as before, we have performed 100 sequential request. The results for this test can be found in Figure 10.

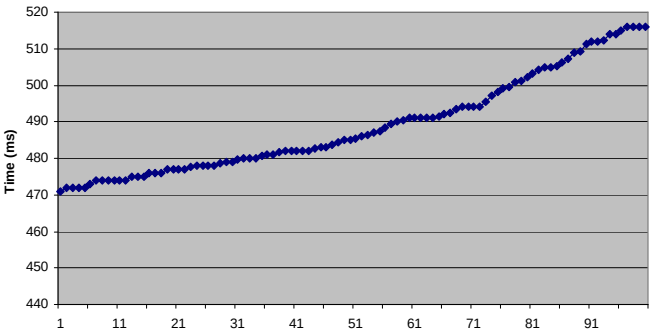


Figure 10. Percentile times for validating a delegation chain

In this case, the PDP takes 489 ms on average to validate the complete delegation chain. This process includes: the retrieval of both policies from their corresponding repositories, the administrative policy from the System Repository and the access control policy from the Delegate Repository;

the harvesting process of the Alice's attributes through the Attribute Requester component; the internal validation from both policies; and, finally, the building and digital signing of the SAML response.

Among these steps, the largest computational load corresponds to the retrieval of the policies from their repositories, taking 196 ms on average. This supposes the 40% of the total time. Thus, and depending on the scenario we are considering, some optimizations could be implemented, e.g., by caching these policies in the PDP, to reduce these times in a real production system.

VII. Related work

The XACML delegation profile specified in [13] is very recent, so there are not many works making use of it. However, the idea of delegation of rights has existed for several years, and there are a lot of works trying to include this feature in different ways. For example, in PKI systems, two initiatives stood out in the use of delegation [23]: SPKI/SDSI defines a standard way for digital certificates whose main purpose is authorization rather than authentication; and X.509 proxy certificates are another way of providing restricted proxying and delegation within a PKI-based authentication system.

In the latest years, authorization systems are making use of XACML because it is standard, and besides it provides an expressive access control language. Before the apparition of the XACML delegation profile, several proposals to add these features to XACML were made. For example, in [24] the authors presented a system permitting controlled policy administration and delegation using XACML in combination with a second access control system. This system is *Delegent*, which has powerful delegation capabilities. But contrary to our proposal, this system does not define how delegates manage the XACML policies.

Another proposal to add dynamic delegation to XACML is presented in [25]. But unlike the XACML delegation profile, this system is based on the delegation of roles instead of the delegation of policy administration. This system defines a validation service whose purpose is to validate a set of credentials for a subject, issued by multiple dynamic attribute authorities from different domains. Finally, it returns a set of valid attributes that are used in the XACML system in the standard way.

VIII. Conclusion and future work

This work shows that although the administrative delegation is a helpful tool for managing policies in complex systems, it also introduces some drawbacks that hinder its use in real existing environments. Therefore, this paper describes an infrastructure that makes use of the administrative delegation in an effective way, thus simplifying the work of both the system administrators and the delegates. On the one hand, the workload of the administrators is reduced by distributing the policy management among the appropriate users in the system, i.e., the delegates. On the other hand, the delegates, who are not concerned about policy management, only have to fill in the appropriate templates to generate these policies in an automatic way.

Currently, as a statement of direction, we pretend to study in depth the management of delegation chains in highly distributed authorization systems.

Acknowledgment

This work has been partially funded by the CENIT Segur@ (Seguridad y Confianza en la Sociedad de la Información) project and the MASTER EU-IST project (IST-2001-34600) within the EC Seventh Framework Programme (FP7). Authors would finally like to thank the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the Fundación Séneca.

References

[1] M. Gil Pérez, G. López, A.F. Gómez Skarmeta, and A. Pasic. "Advanced Policies for the Administrative Delegation in Federated Environments". In *DEPEND'10: Proceedings of the 3rd International Conference on Dependability*, pages 76–82, July 2010.

[2] The ERASMUS Programme Web site, European Commission. http://ec.europa.eu/education/programmes/llp/erasmus 25.05.2011.

[3] Confederation of EU Rectors' Conferences and the Association of European Universities (CRE). "The Bologna Declaration on the European Space for Higher Education: an Explanation", June 1999.

[4] European Higher Education Area (EHEA) Web site 2010-2020. http://www.ehea.info 25.05.2011.

[5] K. Wierenga and S. Winter (main editors). "Inter-NREN Roaming Architecture: Description and Development Items". GÉANT2 JRA5, Deliverable DJ5.1.4, September 2006.

[6] Haka Federation Web site, CSC-IT Center for Science Ltd. http://www.csc.fi/english/institutions/haka 25.05.2011.

[7] SWITCH Federation Web site. http://www.switch.ch/aai 25.05.2011.

[8] M. Sánchez, O. Cánovas, G. López, and A.F. Gómez Skarmeta. "Levels of Assurance and Reauthentication in Federated Environments". In *EuroPKI'08: Proceedings of the 5th European PKI Workshop on Public Key Infrastructure: Theory and Practice*, pages 89–103, June 2008.

[9] E. Rissanen and B.S. Firozabadi. "Administrative Delegation in XACML - Position Paper". Swedish Institute of Computer Science, September 2004.

[10] A. Pasic, J. Bareño, B. Gallego-Nicasio, R. Torres, and D. Fernandez. "Trust and Compliance Management Models in Emerging Outsourcing Environments". In *Software Services for e-World*, volume 341 of *IFIP Advances in Information and Communication Technology*, pages 237–248. November 2010.

[11] The MASTER EU-IST Project (Managing Assurance, Security and Trust for Services). http://www.master-fp7.eu 25.05.2011.

[12] T. Moses (editor). "eXtensible Access Control Markup Language (XACML) Version 2.0". OASIS Standard, February 2005.

[13] E. Rissanen (editor). "XACML v3.0 Administration and Delegation Profile Version 1.0". Committee Specification 01, August 2010.

[14] J. Masa (editor). "SCHema for ACademia (SCHAC): Attribute Definitions for Individual Data Version 1.4.0". Working Draft, March 2009.

[15] Education Roaming (eduroam) Web site, TERENA Association. http://www.eduroam.org 25.05.2011.

[16] D.R. Lopez (main editor). "GÉANT2 Authorisation and Authentication Infrastructure (AAI) Architecture Second Edition". GÉANT 2 JRA5, Deliverable DJ5.2.2,2, April 2007.

[17] Education GÉANT Authorisation Infrastructure (eduGAIN) Web site. http://www.edugain.org 25.05.2011.

[18] O. Gryb. "XACML-Studio (XS) Reference", November 2009. http://xacml-studio.sourceforge.net 25.05.2011.

[19] D. González and A. Neisen. "XACML.NET Version 0.7", March 2005. http://mvpos.sourceforge.net 25.05.2011.

[20] M. Harrah. "Access Control in eXist", September 2009. http://exist-db.org/xacml.html 25.05.2011.

[21] University of Murcia. "UMU-XACML-Editor Version 1.3.2". http://sf.net/projects/umu-xacmleditor 25.05.2011.

[22] M. Kay (editor). "XSL Transformations (XSLT) Version 2.0". W3C Recommendation, January 2007.

[23] M.R. Thompson, A. Essiari, and S. Mudumbai. "Certificate-Based Authorization Policy in a PKI Environment". *ACM Transactions on Information and System Security (TISSEC)*, 6(4):566–588, November 2003.

[24] L. Seitz, E. Rissanen, T. Sandholm, B.S. Firozabadi, and O. Mulmo. "Policy Administration Control and Delegation Using XACML and Delegent". In *GRID'05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 49–54, November 2005.

[25] D.W. Chadwick, S. Otenko, and T.-A. Nguyen. "Adding Support to XACML for Dynamic Delegation of Authority in Multiple Domains". In *CMS'06: Proceedings of the 10th IFIP TC-6 TC-11 International Conference on Communications and Multimedia Security*, pages 67–86, October 2006.

An Adaptive and Dependable Distributed Monitoring Framework

Teemu Kanstrén, Reijo Savola
VTT Technical Research Centre of Finland
Oulu, Finland
{teemu.kanstren,reijo.savola}@vtt.fi

Sammy Haddad, Artur Hecker
Telecom ParisTech
Paris, France
{sammy.haddad,artur.hecker}@enst.fr

***Abstract* — This paper discusses several relevant aspects of performing monitoring in the context of software-intensive systems. The focus is especially on cases where the observed system is distributed, and the monitoring system needs to be secure, dependable and capable of adapting to a number of dynamic scenarios during the system evolution. Based on the analysis of monitoring needs in this type of domain, a set of core requirements for providing a monitoring framework for these domains is defined. To address these requirements, a high-level reference architecture for a monitoring framework is presented. These requirements and reference architecture provide a basis for designing different monitoring systems.**

Keywords – monitoring, framework, security assurance, adaptation, dependability.

I. INTRODUCTION

Collecting data about different aspects relevant to the behaviour, configuration and deployment of a software-intensive system during its lifetime is important for many different purposes. Together with the different partners in the CELTIC BUGYO Beyond project, we have collected and analysed a set of core requirements from the viewpoint of building a monitoring framework (MFW) for continuous monitoring of security assurance-related information. Additionally, we have analysed a set of existing MFWs for different domains with similar requirements. Based on this set of requirements, we present a high-level architecture for a MFW and discuss how it addresses the different requirements. This paper is an extension of our previous work [1], updated with the latest evolution in our research.

There are many potential application domains of operational measurement, including supporting software (SW) quality assurance activities such as testing and debugging that require collecting data about system behaviour for analysis [2]. Quality assurance can also be associated with monitoring different aspects of an operational system, such as the quality of service in a telecommunications network [3], compliance of dynamic systems with their requirements [4] and the security compliance of the system [5]. The monitoring functionality can also be used to provide automated actions such as restarting failed services and sending failure notifications [6]. In the case of some systems, the data collection itself is the main purpose and goal of the system. For example, scientific experiments can require collecting large amounts of data for research purposes [6,7].

Although the measurement systems target different domains, they all share the goal of capturing information (monitoring) about different properties that are important to the functionality and security of the observed system. From the information monitoring perspective, they all thus share the same set of core requirements. In addition, each application of a MFW can also have its own set of specific requirements such as optimization for real-time processing [2] or high-performance capture of large data sets over large networks [6]. In this paper, we focus on the core set of requirements and classify these to five different categories: intrusiveness, security, dependability, scalability and runtime-adaptation. These categories of requirements represent different viewpoints of the monitoring functionality.

First, monitoring typically disturbs the observed system to some extent (commonly referred to as a *probe effect*), affecting its reliability and dependability. Intrusiveness needs to be minimized. Second, in many cases the collected information provides sensitive information about the observed system and its behaviour and thus this information needs to be protected from any unauthorized access (security). Third, to enable best possible use of the information, the MFW design should be highly dependable in order to assure that the data is available even in case of failures in the observed system in order to allow for analysis of the issues based on the captured information. Fourth, it is important for the MFW to be scalable for use in different types of observed systems, where the scale of distribution can vary greatly. As many modern systems evolve during their lifetime or exhibit high dynamics in their structures, the MFW should also support runtime adaptation to account for the measurement needs, the evolution of the observed system and the MFW itself.

We start by defining a core set of requirements that we have synthesized for these types of systems and present high-level reference architecture as a basis for a MFW to address these requirements. The main contribution of this paper is the identification and analysis of core requirements relevant for building a dependable, secure and adaptive distributed monitoring framework, and providing a reference architecture that addresses these requirements.

The rest of this paper is structured as follows. In Section II, we present a number of existing MFWs for different purposes and domains. In Section III, we synthesize a common set of requirements based on the review of requirements in the domains these frameworks have been applied to as well as the requirements we have collected and analysed together with different partners in the BUGYO Beyond project. In Section IV, we present a high-level MFW design to address the requirements. Finally, conclusions end the paper.

II. BACKGROUND AND RELATED WORK

In this section, we present the background concepts related to the discussion in this paper and a number of existing MFW designs for specific domains including a discussion on their relation to our work presented in this paper.

A. Background Concepts

Runtime monitoring is defined here as the act of collecting information about a system during its operation. A MFW is a system that performs the collection (and possible processing) of this information and provides it to interested clients who make use of this information. The raw information is typically detailed in nature and thus is mainly used by other SW components to perform further processing of the data, such as visualizations for human users or issuing system control commands for automated adaptation based on algorithmic analysis. Additionally, a MFW can also provide higher-level events describing observations made of the observed system by local processing nodes (e.g. service failure). These events can also describe information about the MFW itself, such as the evolution of the system infrastructure (e.g. the addition, removal or reconfiguration of a MFW element). These basic concepts apply on different scales, from embedded SW to networked systems of systems, where only the scale of the component and its interconnection mechanisms change.

The basic (measurement) information about a system is captured by a *probe*. A probe is defined here as a SW or hardware (HW) component capable of performing a specific measurement on the observed system. These can be either commercial-off-the-shelf (COTS) or custom-made components. These probes are linked to the MFW to provide the monitoring data to be processed by the MFW components and its client applications. The MFW implements the infrastructure to capture the data over the different observed system elements. In different types of systems, the MFW infrastructure is thus also different, such as spread over the network in distributed systems or distributed over the components of an embedded SW. In this paper, we focus on monitoring of distributed systems, although we note that the principles can for the most part be applied at different scales provided that the system is designed using proper architectural properties (e.g. see [8] for embedded SW).

Specific types of probes can be identified and used depending on the needs of systems. For example, Wang et al. [4] define four types of probes:

- Instrumented probes with analysis – probes embedded in a system which process raw data before outputting it.
- Instrumented probes without analysis – probes embedded in a system which directly provide captured raw data.
- Intercepting probes with analysis – external probes (e.g. network analyzer HW) that provide some analysis of the raw data as output.
- Intercepting probes without analysis – external probes that provide the raw data as captured.

Together these probes provide the raw measurement data to fulfil the system measurement needs. A mediator component is then used to provide information access to clients.

B. Related Work

In this section we describe a number of existing measurement frameworks presented in the literature, and discuss their relation to our work presented in this paper.

The current study presents a part of the CELTIC BUGYO Beyond project results. This project follows up on the CELTIC BUGYO Project. During the BUGYO project, a preliminary version of MFW dedicated to security assurance was designed and implemented [5]. The main challenge of this framework was to provide a solution to the problem of data collection related to the security-enforcing mechanisms of the observed system. In this previous work, the measurement architecture was not deeply investigated and it was expected to be installed in an ad hoc manner. We partly extend this work but take a new and more systematic approach to defining the requirements for a MFW and its practical deployment, with a particular focus on addressing the dynamic aspects present in real systems. This MFW was described to consist of three types of components with the following roles. A component called *probe agent* controls a specific type of a probe, a *MUX agent* provides multiplexing of data over subnet boundaries, and a *server agent* handles centralized processing of the monitoring data and interfacing with a client system. The observed data is not sampled at a high frequency or in great amounts and thus they do not optimize the communication infrastructure but rather focus on using XML-based protocol formats. One main weakness of this MFW was the fact that it was not adaptive and could only run on a fixed architecture. For the rest of this paper, we adopt the agent terminology from this previous work and the associated agent types (basic MFW components).

A MFW for capturing large amounts of scientific experimentation data is presented in [7]. It is aimed at capturing massive amounts of data from sensor electronics located next to the Large Hadron Collider (LHC) detectors with high-performance, scalability and dynamic monitoring requirements controlled by a flexible and configurable GUI. The fundamental feature here is the routing of many sorts of data (from simple parameters to histograms or event fragments). They use Common Object Request Broker Architecture (CORBA) [9] as the protocol between the MFW components to support standards-based implementation on different platforms. A separate data stream is used to pass the high-volume data through the MFW system, and a separate channel is used to pass control requests and events related to the MFW. This aims to provide high-performance data capture with specialized data streams.

A similar MFW (called MonALISA) for capturing data about scientific experiments is described in [6]. Its first purpose was the analysis and processing of large-scale data-intensive grid applications. Their targeted challenge is to provide a MFW able to manage monitoring aspects related to storage, networks and a large number of running applications in near real time. The design of MonALISA is inspired by the Jini [10] architecture where each agent in the framework

is described and registered as dynamic services. The MonALISA design is divided into three layers, each with a specific functionality. The first layer provides dynamic registration and discovery of all MFW components. The second layer consists of the monitoring services, where each provides data and events of a defined type, and the others can subscribe to these data types. The third layer is called the proxy layer and it handles the communication of the MFW components over network filters (e.g. firewalls), providing access control to the monitoring data and to the management of the MFW components.

A monitoring framework for Voice-over-IP (VoIP) traffic is presented in [3]. The main goal of this framework is to allow high-speed, real-time and efficient (in terms of CPU use) analysis of communications traffic. In this framework, data, i.e. packets, are collected and processed at the kernel level using a plugin-based server architecture with Session Initiation Protocol (SIP) [11] and Real-Time Transport Protocol (RTP) [12]. A set of filters is defined for the different plugins to define which data has to be processed by each plugin. The plugins can then in turn be configured to process the filtered data according to their functionality. A library of functions is provided to implement common data processing functions and an easy configuration and update mechanism is provided for the plugins.

A MFW for web service requirements is presented in [4]. The data to be monitored is inferred from the web-service specification (WSDL) of the observed system. A set of constraints over the observed data and events is used to describe the allowed values and event sequences. This includes frequency, interval and ordering of events. Some of their probes provide basic values while others process values before passing them forward. Probes are generated based on the WSDL.

A MFW focusing on embedded real-time systems is introduced in [2]. It focuses on high-performance requirements in a single (non-distributed) embedded system. It defines a custom binary protocol and a set of optimized services for capturing high-frequency data from a running system, focusing on minimizing its use of the observed system resources (e.g. memory and CPU load). It provides a set of services to enable the implementation of features to monitor different aspects of the system (e.g. task scheduling).

The GEMOM (Genetic Message Oriented Secure Middleware) EU FP7 Project has developed an adaptive security, resilience and Quality-of-Service (QoS) MFW [13,14] for distributed information systems. The data communication is arranged by a publish/subscribe mechanism, and separate modules handle authentication and authorization functionality. Specific QoS, resilience and security properties can be set for the different data processed by the GEMOM system, allowing for customization of authorization and authentication for different elements.

From the design factors of the abovementioned measurement frameworks, we apply the following MFW design patterns for the purposes of the framework discussed in this study:

- Specific adaptation layer for measurement targets [5].
- Repository of available probes for specific measurement targets [2].
- Simple interface to integrate custom probes into the overall measurement system [2, 3].
- Usage of widely supported protocols [7].
- Configurability of operational probes [7].
- Optimized communications related to expected measurement data communication patterns [5,7].
- Usage of separated dedicated communication channels [7].
- Customization of different aspects of the MFW based on module composition [3].
- Provide a registry that is dynamically updated to reflect available measurements and probes [6].
- Use specific components to handle connection and processing over network subdomains [5,6].
- Describe the measurements with a common set of properties [4,6].
- Optimize the availability and reliability of measurement infrastructure [13].
- Secure the communication data [13].

In the following sections, we describe why these design patterns are relevant for a measurement framework. We also present a reference architecture that takes these and other MFW requirements into account.

III. REQUIREMENTS

This section describes a core set of issues we have identified for the type of monitoring addressed in this paper. Based on the analysis of these issues, a set of requirements for the design of a MFW is presented. These issues have been divided into five different categories: scalability, runtime adaptation, correctness, intrusiveness and security. An overview of these categories and their properties is shown in Figure 1. These categories are described in the following subsections.

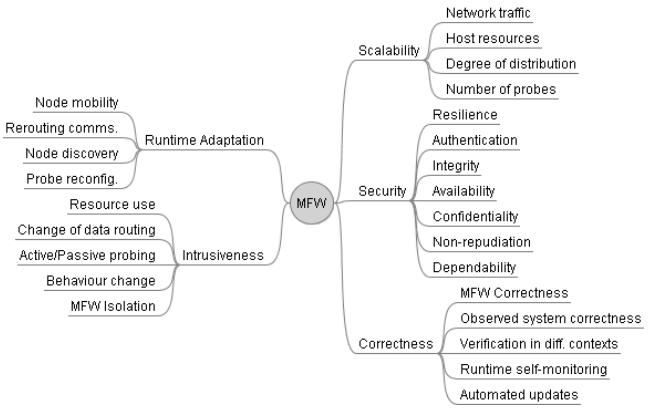


Figure 1. Requirements overview.

A. Scalability

From the basic scalability perspective, the MFW infrastructure needs to be able to handle varying amounts of observed events and data captured from systems with different degrees of distribution. In this case, scalability issues include supporting varying amounts of probes and resource limita-

tions in processing large amounts of data (e.g. CPU and memory use, network bandwidth). For the MFW infrastructure this can require use of alternate communication paths in order to avoid causing network congestion. This can require use of specific MFW components to be deployed at different locations, such as localized processing and multiplexing nodes, routing nodes and probe control nodes.

The scale of distribution of a MFW depends on the nature of the target of measurement. In some cases, it may be possible to perform all measurements from a single centralized location when the measurement nodes already have the ability to provide the required measurement access and functionality remotely, such as over remote Secure Shell (SSH) connections. In more highly distributed systems, a more distributed MFW architecture is needed. In this case, measurements need to be performed for a large number of nodes (possibly divided into subnets), not all of which are reachable from a single centralized (monitoring) location. This requires specific router elements to be deployed to handle connections over the different nodes and subnets as divided by filters (e.g. firewalls).

With regards to the distribution aspect, the addressing scheme used for specifying which measurements can be supported also needs to be considered. Depending on the type of measurement being performed, one may wish to address more than one target in a measurement request. For example, in the security assurance domain, one may request information on the encryption strength of all routers deployed and visible to the MFW. The task of the MFW is then to automatically scale these requests without overloading the network.

As the observed system evolves, the need for such specialized MFW elements may increase, meaning that the addition of these elements needs to be supported in an evolving manner through runtime adaptation.

B. Runtime Adaptation

Modern distributed systems are rarely static in their formation. Even if the system itself supports scaling to different sizes of network infrastructure, it is not enough to simply support deploying these different types of architectures. The system must be able to evolve to support changes in scale while in operation. In today's distributed systems, nodes come and go, and the same nodes can move to different locations in the network. In addition to physical changes in the infrastructure, system reconfigurations can also change the available communication channels. To address these scenarios, the MFW must be able to keep the mapping of its elements up to date and in synch with the changes in the observed system (i.e. which element does a probe measure and which measures it provides), and adapt to changes in the available communication channels between the MFW infrastructure nodes. In order for client applications to build functionality on top of the monitoring data, the MFW must also abstract the changes so that a client can request a specific measure and does not need to worry about the dynamic aspects of evolution such as node mobility.

For routing of data through the MFW infrastructure, various issues need to be considered. If the used communication paths become unavailable, communication channels need to be rerouted. Yet, in many situations and to fulfil the non-intrusiveness objective, the routing in the observed system should not be changed to accommodate the needs of the MFW. For this reason, the MFW must provide adaptation capabilities to address these needs in an optimally adaptive and non-intrusive way.

When a measure has become unavailable but later becomes available again, the MFW needs to reconnect to the corresponding probe and to notify any client applications making use of this information that it has again become available. Similar notifications must be given for all changes in MFW evolution.

As new MFW infrastructure elements are installed, the MFW needs to link them to enable applications to use the data they provide, or for the MFW to use them for the control of the measurement (e.g. data routing). A basic means is manual configuration of all added nodes, but runtime reconfiguration by the MFW also needs to be configured. The optimal case is that the discovery and integration of new deployed MFW elements are automated.

Changes (evolution) in the observed system's infrastructure also affect the monitoring needs. As a basic feature, a MFW must enable the user to deploy changes to the active measurement requests. Additionally, depending on the types of probes deployed, different types of actions are necessary. Embedded probes are assumed to come with the deployed components of the observed system, and require the deployment of a matching MFW component to integrate them with the MFW infrastructure. External probes, on the other hand, can be used to monitor more than one target, and thus, in this case, the MFW needs to reconfigure the (existing) external probe to also monitor the new element that was introduced. For all new probes, there must exist a suitable deployed MFW element to integrate them with the MFW infrastructure.

C. Intrusiveness

Monitoring always has some effect on the target of observation – this is often referred to as the *probe effect*. In the case of the MFW, this translates to the effects that the installation of the MFW components and their use have on the observed system. Installation of the components themselves alone has an effect on the use of some system resources if the components are installed within the observed system itself and/or share some resources with it. In the case of external probes, installation should have only a minor effect on the target of observation unless the installation itself somehow changes the behaviour of the system, such as the routing of messages.

Probes that share resources with the target of observation have an impact on its available resources such as CPU load and memory consumption. These probes are typically embedded probes, running as part of the observed system or on the same HW. The operation of external probes can also affect the observed system in a similar way. For example, routing data through an external probe can add network latency to the observed system. For another example, activating features and tools, such as SFlow [15] and Netflow protocol-

based tools [16], on router ports can be a drain on router CPU cycles. Some of these effects can be minimal yet still cause unwanted effects to, for example, the timing of the system, changing it from the original design specifications. When a probe is more active, for example, sending packets to different network elements, it can invoke untested or unintended behaviour in a system, causing side-effects such as a complete crash or incorrect results.

In the scope of the BUGYO Beyond project, we have identified the following possible probe effects:

- Consumption of hosting device resources (CPU, memory, disk).
- Consumption of network bandwidth.
- Deployment of conflicting SW components, such as shared libraries.
- A probe accessing restricted resources in the system, causing the system to enter a different state. For example, triggering an alarm and a countermeasure.
- A probe accessing system resources (e.g. a file) and causing a lock, while another part of the system is trying to use the same resource.
- Change of privileges required for an agent (or probe) to be able to perform its tasks.
- Opened gates (e.g. ports) to access information in the observed system.
- Changes performed by the probe on the measured system, e.g. by some eXtensible Configuration Checklist Description Format (XCCDF) [17] configurations or benchmarks.
- Active probing of a certain interface of a target of measurement can trigger some unintended features or even crash the whole system, if it has some unintended features linked to this interface.

This set is largely related to our target domain of security assurance, where numerous measurements are done through the file system and with shell scripting. Although most of these are very generic and we see them as being applicable across many domains, some different probe effects may apply in different domains.

The predictability of probe effects is important in assessing which probes to take into use and how. However, combining any arbitrary probes and systems produces unpredictable effects. Even with tested systems and probes, the exact available resources and system configurations can vary and thus it is not possible to provide exact guidelines for any possible probe effects for probe-system combinations. Providing a large-scale analysis of possible probe effects is out of the scope of this paper but it should still be stressed that it is important to take into account the possible probe effects and where possible test for them. Depending on the potential impact, in some cases it may be unfeasible to deploy probes on operational systems with which they have not been tested previously. Knowing the possible probe effects in this case helps in studying them for a specific system and probes. Overall, it can be said that an active probe is more likely to have a higher probe effect than a passive probe.

However, even if the intrusiveness of probes cannot be fully predicted, the MFW should be able to monitor its impact on the observed system and take any measures it can to address these impacts (e.g. re-route communication or start load-sharing components). This requires providing self-monitoring and adaptation features based on specific probes and analysis of the monitoring system properties (e.g. data-stream latency). It also requires providing self-adaptation features such as tuning the probe measurement frequency where observed to be needed.

Even if the implementation and use of probes cannot be constrained, the MFW infrastructure itself, however, can and needs to be designed to be minimally intrusive (i.e. to minimize its probe effect). To achieve this, the different constraints described above need to be considered, as does how they can be minimized in the design and implementation of the different components of the MFW infrastructure. One main goal in avoiding intrusiveness for a system should be the isolation of the MFW from the observed system as much as possible (system independence). Any problems in the MFW (e.g. SW crash) should not affect the observed system, and ideally switching the MFW on or off should have no effect on the observed system. Similarly, the network traffic and any other shared resources of the two entities should be separated as much as possible. Where full separation is not possible, the goal should be to aim for as much separation as possible. Virtual separation should be applied where physical separation is not applicable. When separation is applied, it needs to be applied in both functional and non-functional domains, e.g. for control.

D. Correctness

During the service lifecycle various problems may arise. The correctness viewpoint needs to consider the correctness of both the MFW components and their behaviour. It must also consider their effects on the observed system and how this may affect its correct behaviour. In this regard, the correctness aspect of the MFW is closely related to the intrusiveness viewpoint. Addressing this includes both providing for verification of the behaviour of the whole MFW and all its components before deployment, and monitoring and addressing any problems found in its operational use. Generally, the following three main approaches can be identified from the correctness viewpoint:

- Simplicity in the design in order to minimize the possibility of new problems
- Update mechanisms for the MFW infrastructure
- Testing and verification mechanisms to assure the correctness of the implementations

In practice aiming for simplicity would mean encapsulating more complicated processing of the measurements, control structures and similar properties at a higher level in a separate measurement processing and control layer. However, the choice of how much functionality is incorporated in the MFW infrastructure components (probes and agents) is a choice of tradeoffs in the extent to which it is in the interests of the user for processing to be performed locally (to save on resources such as network bandwidth) or on a dedicated server-agent component that can be hosted on dedicated HW.

Testing and verification are needed to assure the correctness of the implementations in different contexts and configurations (in development and while in operation). Since the infrastructures that are the target of observation can vary greatly and be highly dynamic, the correct functionality of the MFW also needs to be evaluated in different contexts. It is important to verify both the correct functionality of the individual components and their interactions. To support this, the MFW design needs to provide suitable interfaces to ensure required testability features (e.g. as described in [18, 8]).

Even with the best verification and testing techniques, unanticipated situations will arise and failures in the operation of the MFW will be found during its operation in various complex contexts and interactions with different components and systems. To address any errors found during operational use, update mechanisms are needed for the different components of the MFW infrastructure.

A set of specific issues needs to be considered for updating components in an operational system. Three main issues for this are referential transparency, state transfer, and mutual references [19]. Referential transparency refers to updated components not having enough information to notify all related components that are connected to the component in question whether they need to be notified about the update. State transfer refers to the requirement of transferring the state of the component from the previous version to the new updated version. For example, a probe agent may have a state describing its current activity and the state of the connected probe, and all this information needs to be transferred to the new version of the updated component. Mutual references refer to the problem of requiring simultaneous updates on several components due to their mutual dependencies, e.g. during an interface update.

Related to these issues, the updating of the MFW components translates to different requirements for the different components of the MFW infrastructure. This includes both rerouting of communication channels and reconfiguration of any specific functionality, such as the frequency of sampling with a probe. In addition to fixing errors and vulnerabilities, implementation updates are needed in order to enable new features for the different components that are developed to address new requirements for the MFW infrastructure. As many modern systems have high availability requirements, it cannot be assumed that they can be taken down for these updates, and similarly the MFW should keep providing its functionality with minimal interruption.

As the MFW generally collects data from external tools or built-in functions in the observed system, it cannot provide generic support for updating these components external to the MFW. In this regard, the MFW must rely on the user(s) having the possibility to deploy the updates using support provided by the observed system and its components. This can take different forms, such as specific interfaces provided by the updated component (e.g. SSH scripting access) or remote management and deployment tools.

E. Security

Considering the information processed and the functionality provided by the MFW, security issues also need to be adequately considered. Confidentiality, Integrity and Availability (CIA) [20] are the most widely recognized basic dimensions of information security. To minimize the possible impacts of the different security threats on the MFW and the observed system, every means to guard the observed system from abuse of the MFW infrastructure should be taken.

Considering the communication of the measurement data itself, the following viewpoints need to be considered:

- Data protection: Data can be communicated through a non-secure network and should be protected, e.g. through encryption.
- Registration, authentication and non-repudiation: To protect from unauthorized access, each of the new deployed probes needs to be registered and authenticated, and every message between the probes and the aggregation server needs to be authenticated.
- Services availability: To be useful, the monitoring data needs to be available at all times, optimally even when the rest of the system is unavailable, allowing investigation of system failures. To achieve this goal, the measurement infrastructure and aggregation services should be able to communicate data at any time or keep a log file.

The MFW provides continuous monitoring data captured from the observed system. Much of the information that is being collected can describe sensitive details about the observed system, and thus protecting the confidentiality and integrity of this information is as important as the security of the observed system itself.

As the MFW needs to be closely tied to the observed system to enable making the required observations, different factors having a bearing on its impact on the observed system from the security viewpoint must also be considered. The possibility of introducing new vulnerabilities into the observed system by using the MFW and launching attacks against the observed system through the MFW needs to be minimized. In certain attack scenarios, the attacker might even gain unauthorized control of the whole target system through the MFW.

Instrumentation of a given observed system by a continuous in runtime management needs to have adequate security controls in place based on holistic risk-driven analysis. The holistic approach should cope with the resilience of the complete resulting system, including the original observed system, necessary changes to accommodate the MFW functions and the new MFW-specific parts. The ultimate goal is to achieve a resulting system that is more resilient overall than the original system.

To address these issues, the MFW infrastructure must be resilient to malware, relevant vulnerabilities, security attacks and other faults with security effects. For example, Denial-of-Service (DoS) attacks can affect the availability of the observed system through misuse of the MFW and its resource consumption. In addition to considering external threats such as DoS from unauthorized external attackers,

internal threats also need to be considered, such as message flooding attacks or even more harmful attacks by a malicious authorized node.

Overall, an attacker should not be able to use the MFW infrastructure to cause harm to the system under observation by causing it to take excessive resources, gain control of the system, install unauthorized software or otherwise change the system behaviour in an undesired way. The MFW infrastructure should not provide any means of performing attacks against the observed system. Similarly, the MFW infrastructure itself must be protected from attacks and failures. This includes authentication, integrity and availability controls for the communication between the different elements of the MFW infrastructure, proper authorization and data confidentiality and integrity countermeasures. As it is in general not possible to predict and prevent every possible failure or issue in advance, non-repudiation of the data, messages and actions should be at an adequate level to enable audit trail and forensics activities.

An important security-related aspect to be considered is resilience. Even if the observed system is experiencing problems, the MFW should continue its operation as far as possible in order to provide information about the current state of the observed system and to allow for an expert to take corrective actions based on the gathered information. This means the MFW should, if possible, be even more dependable and resilient than the rest of the system components. In some cases, only the latest information may be of interest. However, in other cases the historical data could also be important and in these cases the local nodes need to keep a historical data storage when disconnected from the overall measurement infrastructure.

Overall, the main aspects related to the security viewpoint can be summarized as follows:

- Reduce the chance of using the MFW as a tool to mount attacks against the observed system through, for example, probe registration messages as a DoS attack vector or gaining control of the system.
- Minimize the use of shared resources between the MFW and the observed system to reduce the possibility of it being used as an attack vector.
- Provide sufficient security controls for the measurement infrastructure.
- Use proper authentication and authorization mechanisms for measurement communications.
- Ensure the confidentiality of all processed data as it describes sensitive security-related measurements, both in individual probes and at the overall server level.
- Provide non-repudiation to properly ensure auditability and forensics.
- Ensure any confidential data inside the MFW such as credential and key information for probes (e.g. password).
- Isolate the MFW from the observed system as far as possible.

IV. A Reference Architecture

This section describes a MFW Reference Architecture (RA) and discusses how it addresses the requirements discussed in the previous section. Figure 2 shows a high-level overview of a potential MFW infrastructure with different components. In this architecture, the server-agent is the component that handles the high-level control and processing of the overall measurement infrastructure. The router-agents are the components that take care of transmitting the required data across the network. Probe agents are responsible for controlling different probes at local nodes.

In a practical measurement infrastructure, different compositions of these nodes are to be expected and while the general types of nodes can be described as probe, router and server agents, the exact composition of the individual nodes can also vary. For example, the probe agents need to be adapted to the exact measurement needs, the router agents to the exact routing needs and the server agent to the actual processing needs.

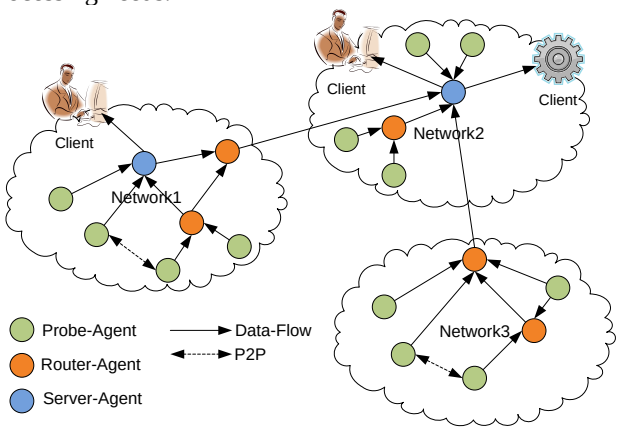


Figure 2. Overall conceptual architecture.

Figure 3 shows a layered view of the same conceptual architecture. It is shown as a set of layers, where each of the layers consists of a set of one or more separate components deployed over the network. Each component in these layers only communicates with the components in the layers directly above and below it. These form a layered architecture as described by Buschmann et al. [27]. This encapsulates the functionality of each layer as a separate, reusable and maintainable piece. A separate communications channel is used as a peer-to-peer (P2P) overlay. While this overlay is not strictly speaking a layer between any of the other layers, but rather stretches over all the other ones, it is shown here separately due to its central role in achieving many of the set requirements. These components will be discussed in more detail next.

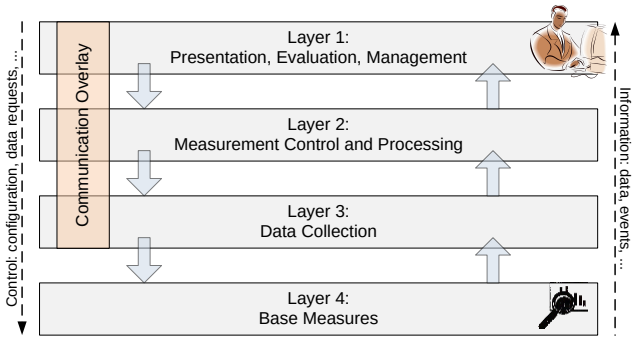


Figure 3. A conceptual layered architecture.

Layer 1 corresponds to the client using the MFW. Typically the communication with a specific client is based on the custom interfaces of the client and needs to be implemented separately. Generally, the functionality of these messages can be described as including the communication of basic measurement values, MFW infrastructure events and passing of measurement requests and configuration messages between the client and the MFW.

Layer 2 handles data processing and configuration control over the MFW infrastructure. For the MFW infrastructure, it handles making notifications of events such as disconnected probes to the client, as well as taking any defined adaptation actions based on observed events. This layer also handles mobility of nodes and keeping the infrastructure model for the client in synch (and abstracting away the dynamic aspects) with the dynamically evolving infrastructure in order to provide correctly over time the required measures to the client.

Layer 3 collects the data from the different probes in the system. It takes the data provided by the probes and communicates this to the MFW control and data processing layer. This layer views the passed data simply as information to be communicated and thus sets no restrictions on it, e.g. whether it has been processed before or is "raw" probe data.

Layer 4 includes the actual probes. It is responsible for handling the base measures for the MFW. The components in this layer are typically not a part of the MFW itself but rather separate components such as commercial off-the-shelf or open-source software components. These are used from the Layer 3 MFW components to perform the actual required measurements and to acquire the requirement measurements.

Considering the links between these four layers, the first layer only communicates with the second layer through a centralized server providing access to the MFW services and measurements. The second layer provides this interface to the first layer and communicates with the distributed nodes of the measurement infrastructure. It issues measurement requests, configuration commands and similar messages to the distributed measurement and router nodes according to the requests from the first layer. It does not see the actual probes performing the measures. The fourth layer only communicates with the third layer by receiving measurement instructions (requests) and configuration data, and providing measurement data in response. This layer is unaware of any other layers and only sees the third layer that serves as the adaptation layer between the probes and the MFW agents. The fourth layer is also the only layer that needs to be directly in touch with the actual observed system in order to provide the measurements. The other layers can be separated to address the isolation requirements. This is where the communication overlay comes in.

The communication overlay is both separate from and interlinked with the different layers. As an overlay it can be separated from the different MFW components by use of standardized network interfaces. At the same time, by binding the MFW components to this interface, it forms a separate, dedicated communication channel for these components. This also allows for handling secure communications through mechanisms in this overlay. The fourth layer is basically separate from this overlay as it communicates not only with the MFW components only but also with probes in the actual observed system. Thus the use of an overlay here allows for minimizing intrusiveness as well.

These layers, the overlay and some generic components are discussed in the following subsections. We start with generic components shared by the different layers, and follow with layer- and overlay-specific components.

A. Component Platform and Automated Updates

Using a component platform as a basis for the MFW provides several advantages. In our implementation, we have used the Open Services Gateway initiative (OSGi) [21] component framework, but other component frameworks with similar functionality can also be used. However, in the rest of this paper we discuss this from the perspective of the OSGi platform. This type of a component platform as a basis provides for loosely coupled services that can be composed in different ways, allowing for easy extension and customization of chosen parts of the MFW. In the following subsections the different layers of the MFW are described as a set of plugins that can be combined in different ways in the different MFW elements to customize and distribute the functionality of the MFW in different ways.

Additionally, OSGi is used as a basis to provide a uniform and automated update mechanism for all components. The goal is to perform updates without interruption of the functionality of the updated components. On a general level, an update can be considered to comprise installing new components, removing existing components or updating existing component implementations. OSGi provides the basic framework for this in allowing for runtime installation and removal of components, although it needs to be extended to make this happen in a distributed fashion.

To support automated updates, each component of the MFW needs to define an interface for reading its internal state so that the new version can replicate the state of the component being replaced. For states that cannot be easily transferred (e.g. data elements whose processing has been started), we use a different replacement strategy. A new version of the component (service) is installed beside the existing one in the same OSGi instance and new messages are routed to the new instance. The old version is removed after it has finished processing all its queued input. This mechanism allows updates of single components. In cases where

the ordering of the data is not important this approach is effective; otherwise queues would need to be used to buffer new messages between the old and new versions [22].

Large-scale updates (e.g. interface changes) are less common but can have a potentially large impact and require a more centralized approach. To support this, a central update manager is needed within the MFW. This keeps track of component versions (which can be queried from the components) and tracks dependencies between components. This enables the management of large-scale updates if there is, for example, an interface change that requires all components to be updated. From the operation correctness viewpoint, these are also more central for the operation of the MFW, as a failure in the update manager will likely cause the complete system to enter a failure state. Thus it also requires more thorough testing and verification. These features of an update system are common to any system embedding such features and we do not go into details in depth here but rather refer to other existing works such as [22]. However, we do note that while much of the existing work on live updates focuses on addressing complex cases, we in practice have experienced that such complex needs in a MFW are rare. The data passed in usually is not sequentially dependent but rather timestamped, thereby simplifying these requirements and the required technical solutions.

In addition to updating elements of the MFW itself, we also expect most probes to require updates, for example, to add new features and address found issues. One option for this would be to use the MFW to perform these updates using a common update interface shared by MFW probe agents. Each probe agent would translate the update requests to a suitable format for the specific probe in question. However, due to the complexity (in organizational policies and technical challenges in heterogeneous systems) of allowing and providing for automated install of arbitrary SW on various systems, we rely on existing tools being used for remote SW deployment to update the (custom and COTS) probes. Thus we do not provide technical solutions in the architecture of the MFW itself for this, but instead rely on existing tools intended for this purpose. However, we identify this as an important aspect of a usable MFW.

However, we rely on updating probe configurations through the probe agents, each of which is expected to have the ability to read and set the configuration of the managed probes. This information is then communicated to the client in order to allow the provision of features for probe configuration management. This is based on the expectation that each configuration can be described with a set of basic data elements (e.g. numerical values and text strings) and each probe agent is capable of describing the probe configuration parameters. Client users are also expected to know enough about their system in order to understand the information required to configure these probes. In special cases where the probes are built to be managed by the MFW itself, they can also be reconfigured by the data processing and control layer.

B. The Communication Overlay and Protocols

In this subsection we discuss the communication aspects of the MFW. This includes both the overlay that is used to provide the separate communication channel for the different MFW components and the protocols used to communicate between the different MFW components over the overlay.

The P2P Overlay

The need for a separate communication overlay has been discussed before. Here, we discuss the use of a P2P-based private, dedicated overlay as a means to solve many of the requirements for runtime adaptation and intrusiveness. The overlay we have used in practice is the Armature P2P overlay, the foundations for which were laid out in [23].

This overlay provides a separate communication channel for the measurements that is dedicated for the MFW. As it is based on Virtual Private Network (VPN) technologies and features discovery of nodes and overlay-layer routing, it provides a separate, secure, adaptive and relatively non-intrusive virtual communication channel. This overlay thus provides us with a virtual (logical) communication channel that can be dedicated for the use of the MFW and deployed alongside the actual nodes of the observed system, which is the type of solution usually needed for practical systems. We call our implementation of this overlay Armature. For better separation, Armature nodes are contained within small virtual machines running embedded-system language interpreters. This addresses many of the intrusiveness and security requirements of the MFW. Besides separation from the system, it also simplifies deployment, as the overlay configuration changes needed for security controls, such as firewalls, in the observed system are the same across the system.

Due to the peer-to-peer nature of the overlay, it also addresses many of the runtime adaptation requirements of the MFW. When deployed, it forms a virtual network among the deployed peers and automatically calculates optimal routes to uphold a robust communication mechanism between the different nodes. When parts of the overlay are separated, they form their own subsystem. When they are re-joined, they will automatically connect to each other and reform new routes as needed. For the reset of the MFW infrastructure (agents) the use of the overlay is simple, as it is simply visible as another network interface on the hosts where it is deployed.

Communication Protocols between the Components

In addition to a communication channel, protocols for passing the data over this channel are needed. An overview of the communication protocols we use is shown in Figure 4, building on the ideas from [5]. A custom communication protocol is needed for each probe as these are assumed to be COTS or custom-made SW components that are not designed with the MFW in mind. It is thus necessary to have an adaptation layer for reading their values and controlling their configurations. This adaptation is handled by the probe agents, consisting of both generic and probe-specific parts. The generic part is shared by all the probe agent implementations and provides a ready implementation of the functionality needed to communicate with the server and router agents over the chosen middleware protocols.

For communications between the different MFW agents, we use a general description of a Message Oriented Middleware (MOM) technology. Using a suitable MOM helps hide the details of communication from different agent implementors and provides all the benefits of MOM use, enabling us to rely on well tested and designed components for this otherwise complex task. It also provides for several choices depending on the specific needs of the target system. For example, one may use open-source protocols such as XML-RPC [24] or Representational State Transfer (REST) [25] web services, or commercial MOM implementations, such as ICE [26], that are heavily optimized for performance and other factors.

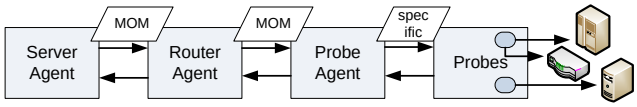


Figure 4. Communication protocol data formats.

C. Layer 4: Base Measurement Layer

The base measurement layer is based upon probes (COTS or custom-made) that are embedded or deployed in the observed system infrastructure in order to collect any kind of data that is considered relevant and can be collected with a probe. A probe agent connects each probe to the MFW and controls the probes. One probe agent can be mapped to one or more probes, and one probe should be mapped to only one probe agent to avoid synchronization issues.

In addition to providing the basic data, this layer also provides events related to the functionality of the probes and probe agents, including the availability of new probe agents and the loss of a probe (becoming non-responsive). It is basically responsible for describing the available measurements from a probe (its characteristics), controlling it as needed and providing the raw measurements from the probe on request. What is supported depends on the types of probes that are available and the features they support.

In our experience, some different considerations need to be taken into account when implementing probes at this level. In some cases, a specific functionality may be needed to perform a measurement of a specific part of the observed system: for example, to read a specific configuration parameter that can only be read programmatically. While we have used a common platform for the development of our MFW agents due to the benefits this brings, as discussed before, we cannot assume that such platforms can be installed on all nodes where measurements need to be performed. Instead we need to be able to make use of what is available and possible on the target of measurement. The probe agent is then a component used as the bridge linking these specific tools and formats to the rest of the MFW.

One generic example here is the use of SSH-based measurement probes. One may have a probe agent capable of creating an SSH connection to a remote host and executing scripts on the target to collect measurement information (such as reading system logs or configuration files, or executing custom probe commands). In this case, the generic output can be, for example, the output of the script execution. Thus in this case the probe is the SSH script executed on the SSH server in the target. The probe agent is a component of the MFW capable of performing these measurements over that SSH.

On the other hand, various constraints need to be considered. In many cases, such as mobile nodes, the address of the target of measurement may vary. Similarly, the availability of the connection to that node may vary. Further, having a separate server of the target of measurement available to respond to queries can be a problem due to the need to have open ports and other similar constraints and intrusiveness aspects. For this reason, a more commonly suitable approach to address these constraints is to make the connection from the target towards the probe agent. In this case, only the probe agent needs to host a server and provide a suitably static address for the collection of the data. The probe agents can then be deployed as needed and will forward the data to the server agent. The server functionality on the probe agent can be different, such as a generic SSH server or a generic Hypertext Transfer Protocol (HTTP) server. However, this type of architecture allows one to deploy any type of a service composition found useful.

D. Layer 3: Data Collection Layer

The data collection layer gathers the data provided by the base measures layer and communicates these to the control and data processing layer. It can also incorporate more advanced features such as processing of the data (e.g. multiplexing) for more efficient communication and smaller network bandwidth use similar to the MFWs described in [5, 6]. In addition to raw probe data, this layer also provides events to the control and data processing layer to describe any observed events in the MFW infrastructure. Looking at Figure 2, all the different agents take some part in the implementation of this layer although this is mainly focused on the router-agents.

In practice, a router agent in this case is a node in the peer-to-peer overlay. These agents handle the relevant parts of the dynamic adaptation, and security features as described before. This layer also handles passing data through (sub-) network boundaries where necessary, through filters such as firewalls. The router agents are basically the mediators of the communication between the server and probe agents, which connect to it by using the overlay interface. For more advanced support, it is possible to build additional agents as separate agents on top of the overlay to support features such as multiplexing of collected data, handling of authentication, authorization and encryption at the subnet level and dealing with network filters such as firewalls and Network Address Translation (NAT) services, similar to [6].

As the overlay sees the data passed through simply as something to be transferred, different strategies to address scalability at the level of data processing at different nodes can be employed without impact on the overall MFW architecture. For example, more of the processing of the base measures can be handled locally by the probe agents, resulting in less data being passed through the MFW infrastruc-

ture. This data can then be handled by a specific functionality in the server-agent. These types of approaches will result in less network traffic and lower likelihood of congestion, distributing some of the processing to the different MFW nodes. The choice of this strategy depends on the needs of the observed system and the MFW.

E. Layer 2: Control and Data Processing Layer

The control and data processing layer provides services for processing the data from the data collection layer and for controlling the MFW infrastructure. This is basically equivalent to the server-agent shown in Figure 2. Specific clients can then be built to access the MFW through the server-agent. In order to support scalability, extensibility and interconnection with different clients of the MFW, we use an architecture based on the blackboard architectural style [27] as illustrated in Figure 5.

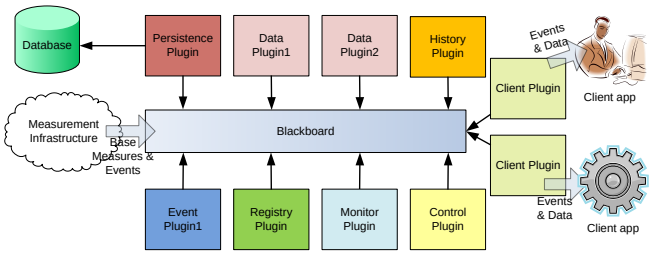


Figure 5. High-level architecture of layer 2.

This type of architecture is actually shared across all the different nodes (agents) of the MFW and not just the server-agent. However, it is described here in terms of the server-agent to provide a concrete example of its application. The goal is to achieve a highly cohesive and decoupled composition of components to support their composition in different ways in different nodes and to address different requirements such as live updates of different parts and verification of functional correctness. To achieve this, all data is passed through the blackboard, whereas messages between the plugins are passed using the OSGi middleware mechanism.

Regarding the data, all measurements, events and client commands are processed through a blackboard component, which provides this data to all registered plugins that have subscribed to this type of data. These plugins can provide additional data to the blackboard, which can further be processed by other plugins. This allows for clear separation of different aspects of data processing, event handling, client and MFW infrastructure communication and other aspects. Again, the plugins are mapped to OSGi services in our implementation. This also provides the means to do updates of specific plugins through the OSGi update mechanism. In the case of message passes where simple data values are not optimal, the bindings are handled dynamically through the OSGi service binding mechanisms. This allows us to address dynamic composition, decoupling and cohesion on different levels.

The control factor in this layer is related to mapping the measurements requested for specific properties of the observed system to the probes of the MFW infrastructure. This includes abstracting all dynamic evolution of the infrastructure(s) in an infrastructure model provided to the client(s). The control factor is also related to responding to events that require taking actions and control over the deployed MFW infrastructure.

In order to support all the requirements described in Section III, different types of functionalities need to be supported. Client-specific data processing functionality can be provided as customized plugins for the blackboard. For example, basic processing functionality can be provided in the form of a plugin that takes instructions from the MFW client that define calculations and threshold values of the monitored data. These can be used to calculate more advanced values from the base measures, and to provide events to the client when a given threshold value for the calculations is exceeded. As the communication is handled through the blackboard, additional processing of the values provided by one of these plugins can be done with another plugin that subscribes to the provided values of the previous plugin.

Customized plugin functionality can be, for example, used to provide specific data to a specific client or to perform custom control and configuration of the observed system based on the observations. Although the needs for different measurement domains may vary, we define a set of basic plugins offering generic services for different purposes. This includes a functionality to store all the data processed through the blackboard to support historical analysis actions, a control plugin to handle the adaptation of the MFW in response to the events observed in its operation (e.g. to configure probe sampling rates) and a registry for handling the abstraction of infrastructure changes to the client. Similar to [3], filters can be attached to any plugin to control the data it processes and to allow for more fine-grained configuration of plugins (e.g. what the persistence plugin stores).

F. Measurement Abstraction

Besides addressing the different needs for runtime adaptation in terms of adaptive communications, the MFW also needs to provide a means for the client to make requests for a specific type of measurement without the need to define explicitly which specific probe will provide it. This abstraction is the role of the control and data processing layer in the MFW (Layer 2). In our case, we have described our solution in our previous work [1] and here we shortly summarize the main points.

Each measurement probe can be identified with a Base Measure Identifier (BM ID) that is composed of a Measure ID coming from BM taxonomy, and a Device ID that identifies which infrastructure object it is measuring. Note that the term Device ID may be misleading at times as the target of measurement here can also be a service and as such hosted on or a subpart of the functionality hosted on a device.

It should be noted that several taxonomies or other approaches can be used as a basis for defining the Measure ID. The actual choice is domain dependent and can even vary inside the domain based on the specific application choice and target inside the domain. For example, in the security assurance domain, we have used as an example the security countermeasures taxonomy from [28], which classifies dif-

ferent measurement targets such as firewalls and intrusion detection systems inside the security assurance domain. Another example inside this same domain is the use of the categories from Common Criteria.

Using the two identifiers we have defined, all measurement results can have Unified Resource Identifiers (URI), or to be exact, Uniform Resource Locators (URL), for example, of type: `MFW://<device_id>/<measure-id`.

A client of the MFW can then make requests for the different types of measurements on different types of targets. The MFW can then provide best-effort measurement results according to the available probes and their characteristics. For more advanced support, the different probes can also be described in terms of different characteristics such as their ability to provide precise results. Such characteristics can only be defined by those who deploy the specific probes, as their interpretations will vary over different probes. However, the MFW can use by default this data as ordinal scale values and let the user define the scales.

Finally, each MFW Uniform Resource Identifier (URI) can then be used as a hyperlink between different measurements. Using this method, it is possible to make BMs dependent on other BMs where this is found useful.

G. Security

As described in Section III.E, security-enforcing mechanisms of the MFW infrastructure need to include user and data authentication, authorization, data confidentiality, data and system integrity, data and system availability, non-repudiation and resilience solutions. This is again supported by the composition of the MFW agents from different services (components). These can be used to compose the agent to support features such as encryption and authentication with specific modules such as described in [9, 10]. This also allows fulfilling the security demands on different scales of distribution. In more distributed architectures this requires decentralization of these features over subnets in the router-agents, while in a centralized version this support can also be centralized at the server-agent level. This can also be handled at the middleware level as provided by the MOM platform (e.g. through the security features of the P2P overlay).

Each MFW component basically has a single user type, which is the higher-level agent (probe → probe agent (→ router agent) → server agent → MFW client) that can issue control over it. Similarly, each one has a single type of a user that can provide monitoring information, which is the lower-level agent (opposite order to the previous one). This knowledge can be used to simplify the required authentication process as only one type of a user needs to be supported at each level, and this user type is always known.

Considering the communication between the different elements of the MFW, each probe agent that controls a probe and provides data to the server-agent needs to register with proper authentication mechanisms before being able to communicate measurement results. Similarly, each data transmission needs to be authenticated to ensure that no false data is provided by unauthorized attackers.

In order to limit the possibilities of attackers using MFW components to perform intrusive actions on a system or as an attack vector, the capabilities of probes and agents should be limited where possible. For example, user accounts can be created to access the required information for the probes, with said users only being authorized to access the required data. To ease management, a roles strategy can be used to enable multi-domain or subnets management where the configuration of the accounts can be replicated without local or specific particularities.

In the following list, we describe some of the technologies and solutions we have applied in addressing these related security requirements.

For achieving integrity of the exchanged data:

- Applying hash methods, which will verify that the received data is the same as the sent data.
- Digital signature methods to provide non-repudiation features.

For ensuring confidentiality of the exchanged data:

- Encrypted communications, communications mechanisms based on public and private key strategies.
- Similarly, applying encryption to all configuration data stored by any MFW node (probe/agent) such as usernames and passwords.

Supporting the availability of the exchanged data:

- Providing features that monitor the availability of the different systems of the infrastructure (agents and cockpit) and that will alert when a device has availability problems. This is related to the self-monitoring features of the MFW.
- The use of data filtering techniques to, for example, reduce the possibility of (DoS) attacks. Specific considerations are needed; for example, when the MFW addresses are mapped for mobility and dynamic aspects (e.g. added or removed element).
- Consequently, the communication protocol used between the probes and aggregation server can propose an alternative management solution, and if historical data is considered to be important, probes should implement the management of local history.

H. Separation of the MFW and the Observed System

As elements of the MFW (probes and probe agents) are installed to measure properties of the observed system, their deployment is bound to have some impact on the observed system. To minimize this intrusiveness, different aspects need to be considered. The first aspect is related to separating the communication of the measurement data and the effects it can have on the observed system. Here we have described the use of the P2P overlay to address this aspect.

Another aspect of isolation is related to deploying elements of the MFW on the same physical host machines as the one observed in the system. In many cases the functionality of the MFW (the agents) needs to be hosted on the observed infrastructure elements. The separation of these two can be addressed by using virtualized infrastructure to host the components of the measurement framework (e.g. Java Virtual Machine (JVM) for OSGi or a complete separate VM).

Finally, intrusiveness can be mitigated with self-monitoring and adaptation. In this case, specific self-monitoring plugins would be deployed at the different nodes (agents). These would monitor for specific problems such as depletion of resources (e.g. CPU, memory, network bandwidth) and adjust their operation accordingly. The specific strategies would require domain-specific tuning according to the criticality of different factors such as the availability of the measurement data and the operational intrusiveness of specific nodes.

I. Patterns and the Reference Architecture

This subsection provides a brief mapping of the MFW patterns listed in Section II. The requirements of the architecture are discussed in more detail in the following section. However, these requirements are also referred to in this section when relevant for the explanation of the patterns. In the following we recall the patterns from Section II and briefly note how they are visible in the proposed architecture.

- Specific adaptation layer for measurement targets [5]. This is the role of layer 3 (probe agents) together with layer 4 (probes).
- Repository of available probes for specific measurement targets [2]. As discussed we identify a set of common mechanisms such as SSH and HTTP probes. In different domains it is further possible to provide specific repositories such as XCCDF in the security assurance domain.
- Simple interface to integrate custom probes into the overall measurement system [2, 3]. This is provided by the split of the probe agent to generic (readily provided) and probe-specific (custom) parts.
- Usage of widely supported protocols [7]. Relying on available and widely developed MOM solutions addresses this.
- Allow configuration of operational probes [7]. This is supported by the interfaces of a probe agent, which can adapt to the probes as best possible.
- Optimize communications related to expected measurement data communication patterns [5, 7]. In our case the assumption is that bandwidth requirements are relatively reasonable, and thus no specific data channels are used. In other cases, this could be supported with the addition of another path.
- Usage of separated dedicated communication channels [7]. This is basically the definition of the P2P overlay we use.
- Allow customizing different aspects of the MFW based on module composition [3]. This is supported by the OSGi platform together with the blackboard-based architecture.
- Provide a registry that is dynamically updated to reflect available measurements and probes [6]. This is supported by layer 2 (server-agent).
- Use specific components to handle connection and processing over network sub-domains [5, 6]. This is transparent to the MFW agents thanks to the P2P overlay.
- Describe the measurements with a common set of properties [4, 6]. This is supported by our measurement abstraction layer.
- Optimize the availability and reliability of measurement infrastructure [13]. Again this is supported by the P2P overlay, which is specifically designed to support these properties, cleanly modularized from the MFW perspective.
- Secure the communication data [13]. The Armature P2P overlay we used handles a large part of the features related to security. Additionally, we provided the guidelines for the security mechanisms we have applied to address this.

V. DISCUSSION

While the mapping of the MFW design patterns was discussed before, this section discusses in more detail how the reference architecture (RA) addresses the different requirements described in Section III. This discussion is structured along the categories of requirements presented in Section III.

The RA supports different scales of distribution by supporting different communication protocols and strategies in the communication layer. The described P2P approach especially allows for a distributed approach. In general, new MFW agent deployment is supported by runtime registration mechanisms. Further, the common plugin architecture for all MFW agents, based on the OSGi component platform, allows for distributing functionality to agents as needed. For example, the use of a common component platform and plugin architecture allows for the deployment of some of the server-agent functionality on probe agent nodes if needed for local processing. This and the ability of server-agents to use any number of different components for data processing support scaling to varying amounts of data. Some specific issues to consider with regards to scalability include the ability to use advanced features such as discovery and registration mechanisms as a means to launch attacks on the MFW itself, and the intrusiveness of monitoring the observed system resource use to launch any measures for dynamic adaptation, such as rerouting.

In the case of many of the runtime adaptation requirements we rely on the work done to address these requirements in the middleware community. The P2P overlay discussed as the communication channel for the RA supports most of these requirements.

The RA uses several approaches for isolating the MFW from the observed system. This includes both the discussed features for the isolation of the network communication and of the SW components of the MFW. The effectiveness of this approach depends on the use of available techniques. For example, using a JVM to host the agents allows some control over the resources it can use (e.g. memory) but can be limited in other regards (e.g. CPU, files). This is a trade-off to consider for different scenarios and may require use of additional advanced techniques (e.g. sandboxing, such as [29]).

As discussed before, addressing probe effects with deployment of various probes in different systems is difficult in general. Testing everything fully in a separate environment

would be ideal, but it can be very expensive and difficult to simulate all the possible combinations of different probes and their environments during a system lifetime. This also applies to the combination of (sub-) systems with other systems. No generic guidelines can be provided for possible probe effects other than checking the probes and agents in operation as far as possible, and considering their properties such as files accessed and resources needed. This also highlights the importance of the isolation aspect.

When using specifically created probes that have been designed into the system from the beginning, intrusiveness is not an issue. Unfortunately, this is not always the case even when designed for, as the future probe requirements are impossible to fully predict. However, the RA aims to deploy only minimal components to the observed system infrastructure, which helps in minimizing its intrusiveness by focusing complex processing in the server-agent. It also focuses on minimizing the intrusiveness in terms of the router-agents by requiring only the deployment of agents for the P2P overlay that share similar deployment requirements.

The basic verification of the correctness of the MFW and its components requires use of testing and verification techniques before deployment. In this part we have to rely on the availability and use of advanced techniques for SW testing and verification. However, we do support this with the interfaces designed for the agents in the form of state transfer (for updates) and configuration access (for reconfiguration). These interfaces form a basis for testability features that can be used to test agent behaviour in various contexts.

Even if we cannot ensure full pre-runtime verification with architectural solutions, the RA addresses runtime verification of different aspects with its self-monitoring features similar to the intrusiveness aspects. This cannot address all possible scenarios but allows for building features to check the correctness of new agents and their functionality in new contexts. This is also supported by the provided automated update mechanisms, which allow for addressing found issues and updating the agents with new features as needed.

The different aspects of security are addressed at the different agents. Specific components are provided to be deployed as services with the MFW agents as needed. These allow for addressing the different requirements related to security such as confidentiality and integrity also at different degrees of distribution. The dependability aspects (including availability and resilience) of security are addressed by the isolation of the MFW infrastructure (similar to the intrusiveness aspects) from the observed system infrastructure as far as possible. This also includes the use of all the security- and scalability-related solutions for handling large amounts of data and unauthorized usage attempts in case of failures in the observed system or malicious usage attempts of the MFW or the observed system.

As for overall security, the MFW is not different from other distributed systems handling sensitive information. We have provided some guidelines regarding the securing of different aspects of the MFW. However, these are generic and applied only to the specific domain of the MFW. More generally, most other approaches for system security in general also apply here and are thus not discussed in detail.

Overall it can be said that different domains set different requirements and in this case some of the issues are more important to address than others. For example, in systems with high performance and large monitoring data streams, it can be useful to optimize monitoring with separate channels as in [6] and optimized data formats as in [2]. The extensibility and customization options provided by the modular reference architecture should provide a good basis for this.

VI. Conclusions and Future Works

This paper presented a core set of requirements for building a secure, dependable and adaptive distributed monitoring framework and reference architecture for addressing these requirements. This is based on both surveying existing approaches to building monitoring frameworks for different domains, and on our current work and experiences in building monitoring frameworks for different domains. The work provides a basis for building other monitoring frameworks that need to address these types of requirements. The presented requirements provide a basis for understanding the different needs of monitoring and how they are related to the domains in which the reader is interested. The reference architecture shows how these can be addressed given the common constraints we present, showing both high-level architectural solutions and practical examples of their implementation. These are topics that are increasingly relevant in many aspects of modern systems, where different runtime adaptation aspects, information collection for decision support, and other aspects need to be supported.

Future work entails describing further experiences of the different implementations and their practical applications in different domains. Future works should also include more systematic consideration of the impacts of monitoring data and analysis on the initial risk analysis that provides input for monitoring as well as their iterative refinements. Events at different levels may also require more attention. Specific future aspects to consider include the emerging new types of infrastructures such as the future internet and cloud-based services where monitoring needs to consider specific challenges posed by the shared infrastructure between different stakeholders, including both infrastructure providers and consumers.

Acknowledgment

The work presented in this paper has been carried out in the CELTIC BUGYO Beyond research project. The authors acknowledge the contributions to the topics discussed in this paper by various project partners.

References

[1] T. Kanstrén and R. Savola, "Definition of Core Requirements and a Reference Architecture for a Dependable, Secure and Adaptive Distributed Monitoring Framework," in *3rd Int'l. Conference on Dependability (DEPEND 2010)*, 2010.

[2] M. Pollari and T. Kanstrén, "A Probe Framework for Monitoring Embedded Real-Time Systems," in *Proc. 4th Int'l. Conf. on Internet Monitoring and Protection (ICIMP 2009)*, Venice/Mestre, Italy, 2009, pp. 109-115.

[3] F. Fusco, F. Huici, L. Deri, and S. Niccolini, "Enabling High-Speed and Extensible Real-Time Communications Monitoring," in *Int'l.*

Symposium on Integrated Network Management, 2009, pp. 343-350.

[4] Q. Wang, J. Shao, F. Deng, Y. Liu, M. Li, J. Han, and H. Mei, "An Online Monitoring Approach for Web Service Requirements," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 4, pp. 338-351, October-December 2009.

[5] E. Bulut, D. Khadraoui, and B. Marquet, "Multi-Agent Based Security Assurance Monitoring System for Telecommunication Infrastructures," in *Proc. Communication, Network and Information Security*, 2007.

[6] I. Legrand et al., "Monitoring and Control of Large Systems with MonALISA," *Communications of the ACM*, vol. 52, no. 9, pp. 49-55, September 2009.

[7] W. Vandelli et al., "Strategies and Tools for ATLAS Online Monitoring," *IEEE Transactions on Nuclear Science*, vol. 54, no. 3, pp. 609-615, June 2007.

[8] T. Kanstrén, "A Study on Design for Testability in Component-Based Embedded Software," in *6th Int'l. Conf. on Software Engineering Research, Management and Applications (SERA 2008)*, Prague, Czech Republic, 2008, pp. 31-38.

[9] Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, 2008

[10] Sun Microsystems, "Jini Connection Technology", 1999.

[11] SIP: Session Initiation Protocol, Internet Engineering Task Force, RFC 3261, 2002.

[12] RTP: A Transport Protocol for Real-Time Applications, Internet Engineering Task Force, RFC 3550, 2003.

[13] R. M. Savola and H. Abie, "Development of Measurable Security for a Distributed Messaging System," *International Journal on Advances in Security*, vol. 2, no. 4, pp. 358-380, 2009.

[14] R. M. Savola and P. Heinonen, "Security-Measurability Enhancing Mechanisms for a Distributed Adaptive Security Monitoring System," in *Proc. 4th Int'l. Conf. on Emerging Security Information, Systems and Technologies (SECURWARE2010)*, Venice/Mestre, Italy, 2010.

[15] sFlow.org – Making the Network Visible. sFlow.org [Accessed: May 21, 2011].

[16] Cisco Systems NetFlow Services Export Version, Internet Engineering Task Force RFC 3954, 2004.

[17] N. Ziring, S.D. Quinn: Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.1.2, U.S. National Institute of Standards and Technology, *NIST Interagency Report 7275* (Draft), 2006.

[18] R. V. Binder, "Design for Testability in Object-Oriented Systems," *Communications of the ACM*, vol. 37, no. 9, pp. 87-101, September 1994.

[19] N. Feng, T. White, and B. Pagurek, "Dynamic evolution of network management software by software hot-swapping," in *Int'l. Symposium on Integrated Network Management (IM2001)*, 2001, pp. 63-76.

[20] D. B. Parker, *Computer Security Management*. Reston, VA, USA: Reston Publishing Company, 1981.

[21] OSGi Alliance: OSGi – The Dynamic Module System for Java. www.osgi.org/Main/HomePage [Accessed May 21, 2011]

[22] Q. Wang, J. Shen, X. Wang, and H. Mei, "A component-based approach to online software evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 18, pp. 181-205, 2006.

[23] A. Hecker and M. Riguidel, "Survivability as a Complementary Operational Security Model for IT Services (position paper)," in *PERADA Workshop*, 2008.

[24] XML-RPC Specification. www.xmlrpc.com [Accessed May 21, 2011].

[25] R.T. Fielding, R.N. Taylor, "Principled Design of the Modern Web Architecture", *ACM Transactions on Internet Technology*, 2(2): 115-150, 2002.

[26] M. Henning, "A New Approach to Object-Oriented Middleware", *IEEE Internet Computing, vol. 8, no. 1, 2004.*

[27] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*: John Wiley & Sons, Inc., 1996.

[28] A. Herzog, N. Shahmehri, and C. Duma, "An Ontology of Information Security," *International Journal of Information Security and Privacy*, vol. 1, no. 4, pp. 1-23, October-December 2007.

[29] Y. Bennet et al., "Native Client: A Sandbox for Portable, Untrusted x86 Native Code," *Communications of the ACM*, vol. 53, no. 1, pp. 91-99, 2010.

Security Test Approach for Automated Detection of Vulnerabilities of SIP-based VoIP Softphones

Christian Schanes, Stefan Taber, Karin Popp, Florian Fankhauser, Thomas Grechenig
Vienna University of Technology
Industrial Software (INSO)
1040 Vienna, Austria
E-mail: christian.schanes,stefan.taber,karin.popp,
florian.fankhauser,thomas.grechenig@inso.tuwien.ac.at

***Abstract*—Voice over Internet Protocol based systems replace phone lines in many scenarios and are in wide use today. Automated security tests of such systems are required to detect implementation and configuration mistakes early and in an efficient way. In this paper we present a plugin for our fuzzer framework fuzzolution to automatically detect security vulnerabilities in Session Initiation Protocol based Voice over Internet Protocol softphones, which are examples for endpoints in such telephone systems. The presented approach automates the interaction with the Graphical User Interface of the softphones during test execution and also observes the behavior of the softphones using multiple metrics. Results of testing two open source softphones by using our fuzzer showed that various unknown vulnerabilities could be identified with the implemented plugin for our fuzzing framework.**

***Keywords*-Software testing; Computer network security; Graphical user interfaces; Internet telephony; Fuzzing.**

I. INTRODUCTION

Voice over IP (VoIP) is in wide use in homes, educational institutions and businesses, extending or replacing Public Switched Telephone Network (PSTN) based phone lines. VoIP provides a way of sending phone calls over Internet Protocol (IP) based networks. This allows the use of one type of wiring for both computers and phones in office buildings, making management simpler and changes in the setup faster. However, moving from separate phone lines to commonly used IP based networks increases the attack surface and by this the risk for attacks. Therefore, a secure VoIP infrastructure is required where all components in the infrastructure are robust against attacks. This also includes the VoIP clients as shown in our previous work in [1].

Today, Session Initiation Protocol (SIP) is a widely used protocol to control communication between two VoIP components like initiation and termination of calls. It was introduced in 1999 by the Internet Engineering Task Force (IETF) in RFC 3261 [2]. It is a stateful text based signaling protocol, which defines two main components for communication: the User Agent (UA) and the Server. A UA can be a soft- or hardphone and initiates or terminates sessions. A Server offers services to UAs, e.g., to register and to relay calls. Phones are reachable by other phones to allow communication and, therefore, a vulnerability within VoIP phones can be remotely exploited by attackers. Our approach shows the possibility to test VoIP phones by using simulated attacks to detect vulnerabilities to make the software more robust.

Fuzzing as a dynamic method to detect vulnerabilities by fault injection was used early by Miller et al. [3], [4] to detect security vulnerabilities in different applications. Since then, fuzzing has become a widely used method to test software robustness and security of different applications [5], [6]. In this paper we focus on testing SIP interfaces of Graphical User Interface (GUI)-based UAs using fuzzing techniques to automatically detect security vulnerabilities by monitoring multiple interfaces of the UAs. For this we present an extension for the fuzzer framework fuzzolution [7].

Thompson [8] defines security failures as side effects of the software which are not specified and make security testing hard. Fuzzing provides a solution for detecting side effects by automatically executing test cases with many data variations based on critical well known attacks and randomly generated values. The introduced fuzzer framework supports the rules presented by Chen and Itho [9] to generate SIP test data. Therefore, an intelligent template based approach [10] with random attack data generation and predefined well known attack values is used. Additionally, with the used state machine it is possible to test different valid and invalid SIP states of softphones.

Automatically triggering GUI events, e.g., accepting or rejecting VoIP calls, is required to test SIP-based softphones because automatic GUI interaction makes it possible to explore a significant portion of the state space. The framework supports all kinds of mouse and key events to interact with the GUI. To improve accuracy of error detection in our approach we monitored multiple interfaces of the GUI-based softphone. One used metric was a GUI monitor on the client side, which allows for detection of error dialogs and changes in the application windows. Other used metrics are log files of softphones, text analysis of the content of open windows on the client side, analyzing the response, monitoring CPU

and memory usage and monitoring the availability of the application by using the network ports of the softphones.

We present the results of our proof of concept approach applied to two open source softphone implementations. We found several previously unknown vulnerabilities. We also tested older releases and found known vulnerabilities, which have already been fixed in newer versions.

The remainder of this paper is structured as follows: Section II discusses related work. Section III introduces the architecture of the test environment and the used fuzzing framework. Section IV gives details about the implementation of test generation and the monitoring of the System Under Test (SUT) to automatically detect errors. The implemented fuzzing framework and detection techniques were evaluated by testing SIP implementations in a VoIP test environment, which are presented in Section V. The paper finishes with a discussion in Section VI and a conclusion and ideas for future work in Section VII.

II. Related Work

Various partly overlapping technologies have been introduced to cover different aspects of VoIP calls, e.g., signaling standards (that take care of the setup of a voice channel). Several signaling technologies are in use today: H.323, Media Gateway Control Protocol (MGCP), SIP as well as proprietary solutions, e.g., InterAsterisk eXchange (IAX) protocol. SIP is probably the most widely adapted protocol [11] and, therefore, in focus of our presented security test approach. As shown by different authors (e.g., [12]–[14]), many attacks against SIP implementations are readily available and conducted in deployed VoIP infrastructures.

Fuzzing is a test technique to find vulnerabilities in different applications [3]–[5] and, therefore, also in SIP-based VoIP applications [15], [16].

Several frameworks have been proposed to address specific aspects of SIP security, for example, the PROTOS SIP Test Suite [15], [17], [18], which has more than 4500 predefined malformed SIP-Invite messages to test the robustness of SIP implementations. However, it only supports stateless testing of SIP phones, which reduces the possibility to test further SIP states like Calling or Ringing. Additionally, PROTOS does not support client testing by triggering GUI actions. Another approach is the one followed by Aitel with SPIKE [19] which introduces the concept of block-based fuzzing. This is based on the fact that protocols are mostly composed of invariants, blocks and variants. SPIKE fills the blocks with fuzzed data and keeps intact the invariants. SPIKE is too low level, so it becomes highly effort-consuming when applied to complex protocols. Banks et al. [20] described SNOOZE, which is a generic fuzzer framework based on user defined scenarios and protocol specifications. Currently only a limited number of primitives to generate malformed data are implemented. However, the authors showed how to use SNOOZE to fuzz SIP implementations and find security vulnerabilities. Another stateful fuzzer is KiF, described by Abdelnur et al. [16], [21]. KiF uses Augmented Backus Naur Form (ABNF) grammar to describe the syntax of messages. KiF automatically generates new crafted messages by using rules defined by the grammar, information of the current protocol state and state tracking information. The authors showed how to test different states of SIP. A drawback is the analysis for detecting errors, which is only based on the responses of the SIP implementations. Nevertheless, with the described approach several vulnerabilities were found. This includes bad input handling and state based vulnerabilities, where the robustness of the implementation failed by using various valid/invalid state paths.

Alrahem et al. [22], [23] presented a fuzzer with the name INTERSTATE. The fuzzer provides the possibility to interact with the GUI of softphones and, therefore, allows, for example, to automatically accept calls. During test execution the fuzzer can send sequences of SIP messages and GUI events to the SUT, which provides the possibility to test a larger range of implemented states of softphones automatically. In contrast to the approach provided in this work, however, the analysis to detect errors is only based on the response and does not consider the behavior of the softphone GUI.

White-box fuzzing is another approach of using internal information of an implementation as, for example, presented by Ganesh et al. [24], by Godefroid et al. [5], [25] or by Neystadt et al. [26]. However, it is not always possible to get the internals of an implementation. For example, if hardphones are being tested, the software is often closed source and only black-box tests are possible. Therefore, we focused on implementing black-box tests for our approach. These can be used for testing a broader number of VoIP clients.

GUI automatization is already used for functional testing of applications as, for example, presented by Feng et al. [24] or by Xiaochun et al. [27]. Bo et al. [28] presented a black-box test approach for mobile phones using Optical Character Recognition (OCR) to analyze the GUI behavior. To improve automatization for testing SIP-based softphones, interaction with the GUI of phones is required as already shown by Alrahem et al. [22], [23]. In our implementation we integrated the automatic GUI interaction. Additionally, we present a basic approach for observing GUI behavior and using the gathered information to reduce the false-negative rate.

III. Architecture for Automated VoIP Softphone Testing

The implemented fuzzer is a generic framework to test and monitor different application interfaces similar to other fuzzers. For testing SIP-based softphones, a template based message generation method was chosen. The fuzzer includes

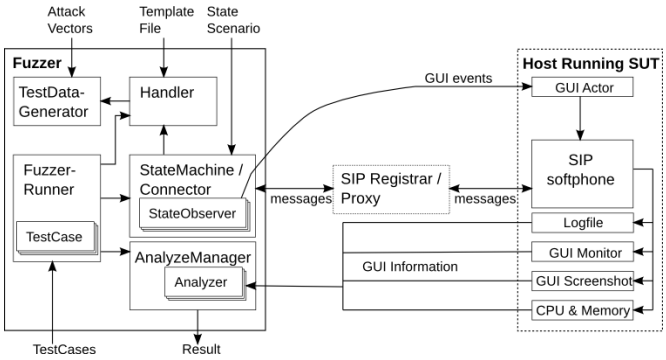


Figure 1. Test Environment of Fuzzer Framework for SIP Softphones

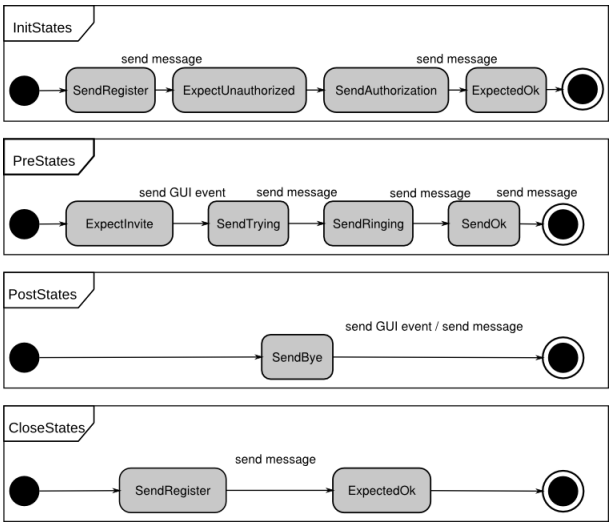


Figure 2. Example of a State Definition

a configurable SIP state machine which allows stateful testing of phones by interacting with the GUI.

Fig. 1 shows the test environment and the interaction between the fuzzer and the SUT. The fuzzer provides the possibility to automatically control and monitor the SUT by its GUI using a GUI Actor and a GUI Monitor. The state machine can trigger GUI events in the SUT using the GUI Actor. Additionally, the GUI Monitor provides information about the GUI to the analyzer which integrates the information to determine pass/fail of tests.

The test environment at the host of the SUT consists of the components softphone (SUT), GUI Actor, the GUI Monitor, the log files produced by the SUT, CPU and memory monitoring and a screenshot component as can be seen in Fig. 1.

A. *Fuzzer Framework*

As framework for executing SIP softphone tests the fuzzer framework fuzzolution [7] was used. This framework provides the possibility to add plugins for specific protocols. We implemented a state based SIP plugin and extended the framework with various analyzers to monitor the behavior of the SUT.

The main parts of the fuzzer framework, as can be seen in Fig. 1, are the `FuzzerRunner`, which controls test execution, a `Handler` to generate the messages to send, the `Connector`, which includes the protocol to communicate with the SUT, the `AnalyzeManager`, which builds the final test result based on all controlled `Analyzers`, and a set of `TestDataGenerators` which provide the used test values.

The implemented `StateMachine` for SIP contains the SIP states and the interaction with the GUI and the SIP interface of the SUT. The framework is independent about stateful or stateless connector implementations. The used `StateMachine` for SIP inherits from the `Connector` type which allows transparent integration into the framework. Based on a configuration file it is possible to configure the various SIP state transitions. Additionally, it supports configuration of invalid state transitions to test the behavior of the SUT by using unspecified transitions. The state machine has a sequence of states to execute during initialization, before sending the test data, and after sending the test data.

The `StateMachine` establishes a connection either directly with the SUT or via a proxy, sends generated SIP messages and receives responses from the proxy or the SUT. Additionally, `StateObserver` can be registered to the `StateMachine`. The `StateMachine` notifies all registered `StateObservers` before and after state transitions. By the usage of the `StateObserver` additional functionality can be executed, e.g., to interact with the GUI of the softphone.

The `Handler` is controlled by the `StateMachine` and is responsible for the generation of valid as well as malformed messages. Additionally, information from the state machine can be used to generate different parts of the message, e.g., the ID. Multiple handler implementations are used for test execution depending on the tested parts of the SUT.

For generating test data various `TestDataGenerators` will be used. It is possible to extend the framework by implementing customized generators. Different implementations are used for generating the test values. They are based on well known attack vectors, random generation and intelligent generation. For more details about generation of test values see Section IV.

For a test case the configuration of state transitions and GUI interactions is called a state scenario. State scenarios are integrated in the fuzzer framework at different execution points: before and after the test run and before and after a single test case. Based on the managed state information, the

StateMachine prepares the SIP softphone, i.e., it brings the softphone into the required SIP state for a test case. For example, a valid registration state scenario can be defined before the test run to register the fuzzer to a SIP registrar and, after the test run, cleanup by de-registering the fuzzer.

Fig. 2 shows an example of a state configuration where as init states the fuzzer registers to a SIP registrar (in the test environment to the proxy). Therefore, the StateMachine sends a Register message and expects receiving an Unauthorized message. The StateMachine resends the message together with a valid authentication header and finally expects an Ok message. After this initial registration for each test case the fuzzer will execute the PreStates before sending the test data and the PostStates after sending it.

A state scenario before a single test case can be used, e.g., to initiate a call from the SUT to the fuzzer by sending a GUI profile to the GUI Actor which initiates the call by executing the GUI profile.

B. Simulating the SIP Registrar

SIP can establish a call directly between two UAs but many SIP phones are registered on a SIP registrar and send all responses to this registrar which acts as proxy and forwards the messages to the destination UA.

When using such a proxy, it is required that the proxy does not filter invalid SIP messages. Therefore, we implemented a specific SIP registrar and proxy as part of the framework which is robust and forwards the manipulated test messages to the SUT without modifications.

The used version of the implemented SIP registrar supports the used SIP scenarios. It accepts all registration requests of any UA and internally stores the registration information, e.g., the username and the address of the UA for further communication.

The SIP registrar supports several parallel UAs. It allows forwarding SIP messages between every registered UAs without further checks. The proxy determines the receiver of the message by the callid or by the username in the To header field or Request header of the incoming message automatically. Additionally, it is possible to manually configure the host and port of the two communicating UAs – in the test setup the fuzzer and the SUT. This also allows to test the To and Request header.

Requests to UAs which are not registered will be handled by sending a Not Found message to the requesting UA. The proxy will respond to every request addressing the proxy with a valid Ok message which occurs often for Option messages.

C. GUI Interaction with the Softphone

The utilization of GUI events is essential to automatically achieve all possible states of the SIP softphone. The provided framework supports the definition of input sequences with SIP messages and GUI events, which are remotely applied to the SUT.

On the side of the SUT an additional service called GUI Actor is running. The GUI Actor is a Java tool, which uses the java.awt.Robot implementation to interact with GUI elements by executing GUI events in the SUT. Using the GUI Actor, automated test execution for GUI-based SUTs is possible.

The fuzzer communicates via network with the GUI Actor using GUI event profiles which are part of state scenarios. A GUI event profile is a sequence of GUI events and an unique identifier to identify the profile. In the current version, mouse and keyboard events are supported. Delays in the sequence can also be configured. This is, e.g., required to wait for opening dialogs in the application before interacting with the GUI.

During initialization the fuzzer registers all required GUI event profiles with the unique identifier at the GUI Actor running in the SUT. For the profiles simple text files with key/value pairs are used. The key is the action, e.g., left mouse click or key press, and as value specific parameters for the action are used, e.g., coordinates for clicking or specific keys to press.

The GUI Actor can handle several profiles simultaneously. By using the profile identifier the fuzzer can execute every registered profile by sending the execute command and the profile identifier to the GUI Actor.

D. Observing System Under Test Behavior

To reduce the false-negative rate, the introduced framework uses multiple metrics of the SUT to build the final test result. For this the fuzzer framework supports the combination of various Analyzers.

One analyzer was used to verify if the arrived response from the SUT is valid (e.g., the expected message for this state transition) and if it contains specific keywords indicating a failure, e.g., Exception or Error. A second analyzer monitors the SUT by verifying if the port of the application is available after executing a test. If the port is not available it indicates a possible crash of the application. Both analyzers can be used for black-box tests without access to the host where the SUT is running.

An analyzer was used to analyze the log files of the SUT and determine important log entries using keywords. For this the log files will be transferred using a Secure SHell (SSH) connection. The analyzer uses only the entries produced during executing the test case by building the delta of the log file before and after the test execution. The analyzer uses the detected log entries and based on a predefined list of weighted critical keywords returns the build probability combining the number of keywords and the associated weight.

Another observed behavior of the SIP softphones is the GUI. Applications visualize error information in different

elements of the GUI, e.g., dialogs. The analyzer monitors the GUI of the SUT and identifies changes of open windows in the host after the test case in comparison to the windows before executing the test case. For all newly opened and closed windows, the analyzer estimates the relevance by using the name of the title and relation of the window to the SUT. The relevance of a window in the SIP softphone is higher than the relevance of other applications in the host. Examples of changes are new open error (or similar) dialogs or disappeared windows, e.g., after a crash of the SUT. Both analyzers require access to the host running the SUT.

In addition to the window structure the framework uses the visualized text information for error detection. For this OCR is used for a screenshot of the SUT to extract the text information displayed on the screen. The screenshot will be taken before and after executing the test case. With OCR the text will be extracted and analyzed using a list of critical keywords. The screenshot analyzer can, therefore, also detect errors displayed directly in the main window without opening a new sub window.

The framework also analyzes CPU and memory usage of the softphone process in the SUT to detect heavy load during/after executing a test which could indicate a Denial of Service (DoS) attack.

The GUI Monitor, the screenshot analyzer, softphone log file analyzer and CPU/memory monitor require access to the SUT to start a service to transfer the information to the host running the fuzzer. Both services wait for a connection from the fuzzer and the fuzzer has to trigger the services to get the information.

IV. Framework for Implementing Security Test Cases for SIP-based VoIP Softphones

The fuzzer framework with the SIP plugin provides various possibilities to configure tests for SIP implementations and especially for GUI-based softphones. Within the fuzzer framework the following notation is used for the test configuration which also builds the further structure within this section:

Test Scenario: the scenario is a specific sequence of state transitions, e.g., send an Invite message to the SUT. We executed the fuzzer for each scenario.

Test Case: we define the different fields within a single message as test case. These fields will be replaced by generated values, e.g., an Invite message has several fields and each field indicates one test case.

Data Variation: data variations are the different values used for filling the fields in test cases. We use many data variations for each test case.

Chen and Itho [9] describe five rules that can break robustness of SIP-based VoIP systems by manipulated messages:

- Incorrect Grammar
- Oversized Field Values

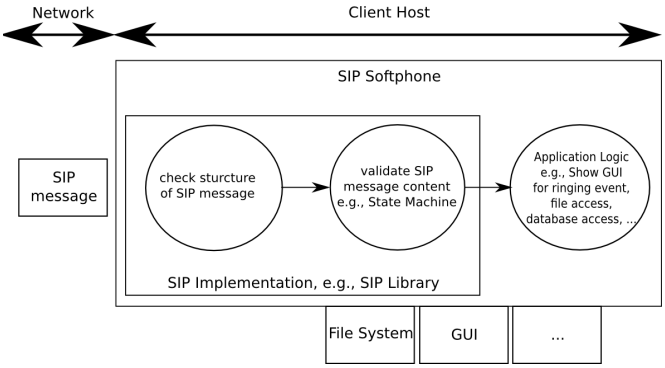


Figure 3. Generic SIP Message Parsing and Processing by SIP Implementations

- Invalid Message or Field Name
- Redundant or Repetitive Header Field
- Invalid Semantic

All of these five rules are considered for testing the SIP softphones with the defined test scenarios, test cases and data variations in our test setup.

Analyzing the attack surface of an application is important for conducting security tests. The attack surface shows possible interfaces with associated risks for exploiting vulnerabilities in the interfaces. Fig. 3 shows a general attack surface for SIP-based softphones. For the current work we considered remote attacks using the network interface of the application. The figure also shows relevant modules within the application which are involved in processing SIP messages.

A. Test Scenarios for SIP

The fuzzer provides the possibility to configure different test scenarios for testing SIP-based VoIP softphones. Each scenario is a single test run of the fuzzer tool. In the current version of the implementation we defined different scenarios to show that the proof of concept of GUI interaction and monitoring of the SUT was working. The analysis is based on various defined SIP scenarios as presented by Johnston et al. in RFC 3665 [29].

Fig. 4 presents the main test scenarios used for the test execution. The first scenario presented is stateless and has already been tested repeatedly using different fuzzers as, for example, PROTOS. The fuzzer sends malformed Invite and malformed Cancel messages to the SUT. Many DoS attacks are based on such a scenario, because no additional conditions are required, e.g., a valid SIP account, to send a message to the SUT. The second scenario represents a typical SIP call flow where the fuzzer calls the UA (SUT). The call flow is a RFC conform state transition. The fuzzer uses the final ACK message to construct test messages. In comparison to the second scenario, in the third scenario the SUT initiates the call. The fuzzer triggers this by the GUI Actor and the SUT calls the fuzzer, which replies with

a Trying, Ringing and Ok. The fuzzer terminates the call without waiting for the required ACK message from the SUT by sending a Bye message. Each message sent by the fuzzer in this scenario is used for the test execution to include the test data. This scenario tests the robustness during the call initiation phase of the UA. The fuzzer initiates a call in the fourth scenario. After receiving the Ok message from the SUT the fuzzer sends a manipulated Cancel message. Similar to the third scenario the SUT calls the fuzzer, which responds with a valid Trying and Ringing. After that the fuzzer sends a malformed Unauthorized message. This Unauthorized message is not expected by the SUT. Therefore, we test how the softphone deals with unexpected and manipulated messages.

For all scenarios except the first one, access to the GUI of the SUT is required for automatic testing in order to initiate, accept and close calls.

The fuzzer supports valid scenarios with RFC conform state transitions and invalid scenarios. Valid scenarios are required to bypass message validation functionality to inject the test values in the application logic as can be seen in Fig. 3. If the validation is not successful the message will be rejected and testing of the application logic is not possible. With invalid scenarios the validation logic within the SIP softphone can be tested.

B. Test Case Design for SIP

Test cases in the fuzzer framework are messages which are part of a test scenario. For the tests a template based approach was used. The template defines the message structure and defines placeholders which indicate the single test cases within a test scenario. The `Handler` within the fuzzer framework is responsible to prepare the final message by replacing the placeholder of the current test case with the attack value of the data variation. All other placeholders are filled by using valid default values which can be predefined static values or generated values, e.g., unique identifiers or timestamps. The `Handler` must also support handling specific values, e.g., the `Content-Length` header. Otherwise the UA will eventually drop parts of messages.

The messages are constructed according to the SIP RFC to provide a valid structure and only use test data within the fields to test application logic as can be seen in Fig. 3. Moreover, tests with invalid structures are used, e.g., randomly modified orders of SIP headers, duplicate, randomly injected characters or invalid header fields. These tests are required to test robustness issues of the SIP message validation functionality.

The fuzzer framework supports cascaded `Handler` configurations, e.g., use the template handler to prepare the message and afterwards use a handler to manipulate message structure.

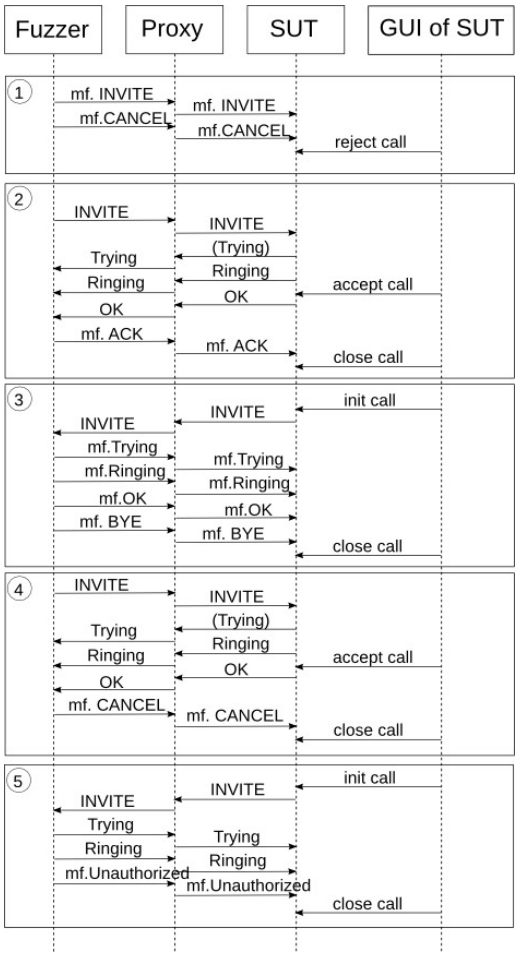


Figure 4. Definition of Used Test Scenarios for SIP Softphone

C. Automated Test Data Generation

Several test cases with several thousand data variations are defined based on the rules mentioned by Chen and Itho [9]. For the applied fault injection approach well known attack vectors and randomly generated data can be used. For the tests an optimized list with about 1200 known critical values as attack vectors are used. This includes various security attacks, e.g., buffer overflows, Structured Query Language (SQL) injections and path traversal attacks.

Additionally, generators are used to generate test data. The fuzzer framework allows integration of additional generator implementations to test specific values. For the test execution randomly generated bytes with random length are used.

D. Automated Error Detection

The fuzzer framework uses multiple analyzers to determine pass/fail of tests. Different analyzers are used for testing the softphones and with each analyzer it is possible to detect different vulnerabilities. For coordinating the results of these analyzers an `AnalyzeManager` is used which collects the results of all analyzers to calculate the final result

for one data variation by summing up the results. In this step it is also possible to change the weight of the results based on predefined rules.

Before starting with the tests the fuzzer initiates a learning phase. In this phase valid data variations are used to determine how the application behaves with valid requests so that differences can be detected when using attacks. During the learning phase, analyzers store information about the found error indicators, so that they can be seen as normal behavior of the SUT and not used for further error detection. For example, the log file analyzer stores the information of how many times each keyword can be found for a valid request in the learning phase. The analyzer only uses keywords for the test execution which occur more often than the initially learned number of found keywords in the log file. This is required because depending on the error handling keywords like ”Exception” can also be included in the log file for valid requests.

Several analyzers are used to monitor the SUT as described in Section III-D. The port analyzer checks if the port, where the SIP softphone is listening, is still available by connecting to the port. The weight of this analyzer is very high and indicates that an error was detected certainly when the application cannot be reached any more.

The GUI analyzer uses the list of open windows in the SUT. We implemented the GUI Monitor prototype running on the SUT. The GUI analyzer retrieves the required information by a network connection from the GUI Monitor which uses the `xwininfo` utility for X to read window information. In an operating system a tree with one root window is available and every other window has exactly one parent window. Our GUI Monitor parses the output from `xwininfo` and stores the information in a tree structure. For each test case this is done before and after executing the test case.

In addition to the list of open windows we used an OCR analyzer which extracts the displayed text from a screenshot of the host desktop and uses a list of keywords for error detection. As OCR engine we used the `tesseract` command line tool. For taking the screenshot a service is running on the SUT. The analyzer requests the image using a network connection and extracts the text using OCR. Example keywords used for the test execution are: ”exception”, ”error”, ”wrong”, ”fault”, ”failure” or ”debug”.

With a response analyzer the fuzzer analyzes the response of the application and checks if certain keywords can be found. The keywords are stored in a configuration file as pairs of keyword and weight of the keyword. The analyzer checks for each keyword if it is included in the response and sums up all weights of the found keywords.

The log file analyzer is very similar to the response analyzer, but uses log files instead of responses. The analyzer determines the delta of a log file to use only log entries logged during execution of the current test case. Example keywords used for the test execution are: ”NullPointerException”, ”IOException”, ”Exception”, ”Bad”, ”Missing”, ”error”, ”fatal” or ”segmentation fault”.

The CPU and memory analyzer monitors the softphone process in the SUT. During the learning phase the analyzer determines CPU and memory usage for valid requests. For each test the analyzer determines the variance to the learned behavior. Depending on the implementation increased CPU or memory usage is possible due to internal reorganization activities, e.g., reorganize or cleanup a cache.

The `AnalyzeManager` gets the probability of a detected error from each analyzer. Each analyzer has a configured weight and based on defined rules the analyzer manager combines the final result using the computed probability of the analyzers and the weights of the analyzers for the final result.

Additionally, the analyze manager performs a final grouping of the result which supports possible required manual checks and retests. The grouping is based on the assumption that test cases with the same probability value have also similar log file entries, similar response values, similar GUI behavior etc. The group contains the list of test case IDs and the detected problem, e.g., a specific exception in the log file. For manual verification of the result to determine if the reported error is a true error, often only one test case per group has to be retested. Furthermore, the higher the result value of the group the more likely it is, that the reported error is a true one. Currently the implementation determines pass/fail based on a configured threshold of the combined probability.

V. Results of the Automated Test Approach

The fuzzing framework fuzzolution was used to test two SIP-based VoIP softphones. This section describes the softphones (SUT), the used VoIP test setup and the detected vulnerabilities. A description of the evaluation of the approach is also included.

For the test execution the test scenarios defined in Section IV were used. With each scenario various unknown vulnerabilities could be identified. These include DoS, memory corruption, improper validation of array index, use of uninitialized variables and a kind of Cross Site Scripting (XSS) vulnerability in both tested softphones.

The test execution showed that every implemented analyzer has its strengths and weaknesses. The combination of the analyzers significantly improves the overall test results.

A. VoIP Test Setup

For the test execution the VoIP test environment as described in [30] has been used. The environment supports various requirements for executing security tests:

Flexibility: Different virtual images provide basic functions, e.g., a time server, a SIP registrar, Domain Name System (DNS) server etc. which can be used as a basis

for the test setup. This way we could concentrate on configuring the relevant aspects for security testing the VoIP softphones with the fuzzing framework which was the objective of this work.

Scalability: The VoIP test environment is scalable. Fuzzing softphones needs a lot of resources to test multiple attack vectors. Especially for GUI-based applications execution time is increased because display of the GUI element and executing the action is asynchronous. To reduce execution time, multiple virtual client images were executed in parallel for the conducted tests. This way the execution time for multiple test scenarios could be significantly reduced.

Easy Analysis: A version control system is implemented to provide an easy mechanism for managing, e.g., the tested software versions, used configuration files, log files or network dumps. This helps analyzing security failures found during the security test process and provides a mechanism for retesting test cases in case of found security vulnerabilities.

Automation: Many aspects of the test process can be automated. Therefore, the needed test setup for executing the security tests can be quickly achieved when doing different tests.

Moreover, we used the capability of the test environment of capturing the network traffic. This allowed detailed analysis of the transferred messages.

During test execution two instances of the two tested VoIP softphones (see Section V-B) were used. The whole process of starting logging and network dumps, starting the softphones, GUI Actor/Monitor, the fuzzer etc. was automated by using different scripts. After the test execution the relevant data (e.g., log files of the VoIP softphones, output of the fuzzer framework) were automatically collected and added to the version control system.

For correlating and analyzing multiple events and log files a common time source was needed. This was achieved by using the time server provided by the test environment. This way timestamped events in the test environment, even if produced in different parts of the test environment (e.g., fuzzer and SUT), could be combined easily and, e.g., fuzzed input strings could be correlated to malfunction of the softphones.

The management interface of the test environment was accessible remotely. This enabled starting and stopping test cases efficiently.

B. Tested SIP Softphones

Two different softphones were tested. The first softphone is QuteCom [31]. It is an open source SIP implementation written in Python, C and C++. We mainly tested the version 2.2 revg-20100116203101 with our fuzzing framework presented in [1]. With the extended framework we tested version QuteCom 2.2 revg-20101103220243. The second softphone is SIP Communicator [32]. It is an open source Java VoIP and instant messaging client. For the work presented in [1] we tested the version 1.0-alpha3-nightly.build.2351. For the extended framework presented in this paper we tested SIP Communicator 1.0-alpha6-nightly.build.3189.

Both applications are available for Windows and Unix platforms. Linux Ubuntu was used as the host system running the SUT for the test execution. Tests during development of the fuzzer framework also showed the possibility to identify vulnerabilities for a previous version of QuteCom on Windows (without using the GUI monitoring) where QuteCom failed to check the content-length SIP header. QuteCom crashed every time when the content-length was less than the real content-length. When the current version of QuteCom was tested this vulnerability had already been fixed. In a future work the GUI interaction and monitoring will also be implemented for Windows operating systems.

C. Detailed Test Results

By implementing the fuzzer framework it was possible to detect vulnerabilities for both tested softphones. The vulnerabilities were reported to the developers. A DoS vulnerability in SIP Communicator could be identified. SIP Communicator had implemented a fixed source port range between 5000 and 6000. It does not reuse port numbers and, therefore, for each connection a new port number will be used. After 1000 used ports no additional calls could be handled by the application. The error was detected by the GUI analyzer, because the application showed an error dialog but did not crash. Therefore, the process of the application is available and it could not be detected by monitoring the processes. Monitoring the availability of the port of the SUT will not register a fail either, because the problem was only for connections from the SUT to another host. Another solution to detect such an error is by using a valid use case which will fail if the application is not available anymore.

The problem was already fixed in the current version of SIP Communicator but the fuzzer identified a similar bug in the new SIP Communicator version. Again, only a range of ports is available and with many open ports the application logs an exception to the log file. The log file analyzer detected this problem. Moreover, it was also detected by the GUI analyzer because SIP Communicator opened additional dialogs. The vulnerabilities were detected, because due to the automation of the test process it was possible to send many requests, which causes these problem.

Another problem in SIP Communicator could be identified during the monitoring of the application log files. The application logs many `java.lang.NullPointerException` for different manipulated calls. This problem could not be identified using the GUI monitoring, because the GUI does not display the error information. We also detected several

`java.lang.ArrayIndexOutOfBoundsException` errors. The log analyzer reported this as error because the keywords are contained in the log messages. Additionally, also the GUI monitoring detected the error because it was a problem with open dialogs which are not closed anymore.

Additional tests of SIP Communicator showed further crashes of the application. The GUI analyzer has identified the problem, because the process of the application terminated and, therefore, all windows of the application disappeared. This was a change of windows, which the GUI analyzer interpreted as an error in the application and, therefore, the test case failed. Such a termination of the process could also be identified by monitoring the process or by trying to connect to the port of the application. Retests of single test cases could not reproduce the problem. For future development the fuzzer should automatically retest different combinations of test cases of a test run to automatically identify such problems.

The fuzzer framework identified a type of XSS vulnerability because SIP Communicator uses `javax.swing.JOptionPane` for constructing error dialogs, which uses `javax.swing.JLabel` to render the text. By directly sending an Invite message to the SUT with a manipulated From SIP header it was possible to inject HTML code, which was displayed by the JLabel. This did not allow the injection of JavaScript code but it was possible to create a connection to a remote host and download an image by using an `<img>` element. In combination with a Cross Site Request Forgery attack this could compromise the internal network of the user. This error was detected by the GUI Monitor because by the injection of Code the displayed dialog has a different size and, therefore, was not closed by the used GUI Actor.

QuteCom is also affected by a DoS vulnerability, which crashed the application. The SUT calls the fuzzer, the fuzzer accepts the call and sends a Bye message to terminate the call. Additionally, the fuzzer uses the GUI Actor to click on the terminate button in the SUT, which causes the crash in the application. The problem was detected by the GUI analyzer and the port analyzer.

Several memory corruption vulnerabilities could be identified for QuteCom, which stop the application with a segmentation fault. One error was identified with multiple parallel calls in Ringing state. Closing the ringing dialogs crashed the application. Another error could be identified by opening several parallel calls. QuteCom shows the error message ”insert number for forwarding”. If the fuzzer triggers the logout button the application crashes. The crashes were detected by monitoring the port of the SUT and by the GUI Monitor because if the application crashed the list of open windows changed. This vulnerability showed that in future tests additional scenarios for testing parallel interaction should be defined. During development we also detected a DoS vulnerability if two separate accounts are configured. QuteCom regularly sends Option messages which the used SIP Registrar answers with an Ok message. QuteCom crashes directly with this behavior. For future work it will be required to integrate further scenarios including several registrations and parallel tests.

QuteCom crashed after sending a message with a manipulated Length field. The error was detected by using the port analyzer because the port is not available anymore after the crash of the application, GUI Monitor because the window of the softphone closed and CPU usage because the process terminates.

During testing SIP Communicator we also detected a problem in the GNOME implementation. The applet displaying an incoming call caused high CPU usage in the system. We detected the problem because the tests take quite longer than other tests. This problem showed that for future tests, in addition to monitoring the CPU and memory usage of the softphone process, also the CPU and memory of the system should be analyzed because a misbehavior of the softphone could also lead to an unstable system.

VI. Discussion

Evaluating the performance of a fuzzer is a difficult task, because only the detection of new vulnerabilities is measurable. Without a benchmark with known vulnerabilities, further analysis is not possible. The fuzzer detected vulnerabilities, which were analyzed manually to evaluate the result of the fuzzer.

The results showed that the implemented fuzzer framework could identify vulnerabilities in SIP-based softphones. The combination of multiple analyzers for a test execution is essential to automatically detect vulnerabilities and reduce the false-negative rate. The implemented combination of the single analyzer results with the initial learning phase and the weight for each analyzer improved error detection compared to the results presented in [1].

Multiple detected vulnerabilities showed that interaction with the SIP softphone via its GUI is essential to automatically execute security tests. Some problems could only be identified by interacting with the GUI.

The analyzers detected different vulnerabilities. The accuracies of analyzers are different depending on the measured property. The analyzer monitoring the port detects failures with high accuracy. Only network problems caused false-positives, e.g., if a firewall blocks required ports.

Analyzing the responses based on a keyword table, i.e., verifying if a response contains specific keywords indicating an error, has not produced reasonable results. The quality of using this information for automated testing strongly depends on the implemented error handling. In the tested applications no relevant information for the analysis was sent in the SIP response.

Analyzing GUI behavior showed that this can be an important information for detecting vulnerabilities. Some

of the identified vulnerabilities were only detectable by such an approach. The used OCR analyzer did not detect vulnerabilities but the produced screenshots were helpful for verifying the produced test results. For future work the OCR analyzer should be improved to increase detection of the characters within the image.

In the work presented in [1] Asterisk was used as a proxy in the test environment. We encountered problems with some messages, where Asterisk filtered messages and, therefore, Asterisk was the test target and not the intended SUT anymore. For the tests presented in this work we implemented a special proxy, one which forwards fuzzed data without modification.

By analyzing the fuzzer output, we identified significant differences in the execution time of test cases within a test run. We further investigated this finding, but could not reproduce the results. By using timing as an aspect of an analyzer during automated tests, it is required to use special test environments which do not falsify timing properties. In the used environment this requirement was not fulfilled.

For future work, the performance of the fuzzer should be improved. Especially the GUI interaction should be enhanced, because currently it is required to define delays in order to allow the application to open windows, e.g., to start a call. Minimizing the delay could improve the runtime performance of test executions. For optimization we used a test setup where multiple instances of the softphones are tested parallel by the fuzzer. However, our focus for the current implementation was the automatization of vulnerability detection as opposed to minimizing execution time.

VII. Conclusion and Future Work

This paper presents an automated security test approach, which increases the security of VoIP communication by identifying vulnerabilities in GUI-based SIP softphones. The provided state-based extension of the fuzzer framework fuzzolution allows for testing SIP states by sending SIP messages and GUI actions to control the softphone, e.g., initiate calls.

Multiple analyzers are integrated to automatically monitor the behavior of the SUT to detect vulnerabilities of VoIP softphone implementations. The responses of the SUT are analyzed and the port of the application is monitored. Additionally, the framework uses the log files of the softphones and observes the GUI behavior of the SUT, which utilizes changes of window states to determine errors, e.g., newly opened dialogs or closed windows. The CPU and memory usage of the process are monitored to detect extensive usage after sending the test attacks. The implemented OCR analyzer could not find an error because the displayed text was not recognized correctly. However, the screenshots taken by the OCR analyzer are helpful for further manual checks.

Messages are sent directly from the fuzzer to the SUT for some test scenarios and for other scenarios a proxy is used to send the messages. Instead of using Asterisk as done for previous tests we implemented a special and robust SIP registrar and proxy for the tests. Asterisk filtered some of the invalid messages. The current used implementation does not filter messages.

With the implemented fuzzer framework, two open source SIP-based softphones were tested and various security vulnerabilities could be identified, e.g., DoS, memory corruption, improper validation of array index, use of uninitialized variables and a kind of XSS. The amount of vulnerabilities found in previous tests showed that further extensive security tests with additional scenarios and variations are required for the softphone applications. The current executed tests also uncovered many security problems of the implementations. As further work the possible SIP states should be configured automatically to get the whole SIP state space for tests.

The previous version of the fuzzer framework produced many false-positive results. In the current version the accuracy of the fuzzer test result was increased by improving the analyzers on the one hand and on the other hand by using a combination of the single analyzer results. To reduce required time for manual analysis many additional functionality has been implemented, e.g., screenshot capturing or grouping of test results based on the analyzer outputs.

The test execution was done on Linux systems. For future versions it is required to implement GUI interaction and observation for additional operating systems.

With the presented approach various vulnerabilities of SIP-based softphone implementations could be identified. The results show that GUI interaction and observation is required to automatically test for security vulnerabilities of softphone applications efficiently.

References

[1] S. Taber, C. Schanes, C. Hlauschek, F. Fankhauser, and T. Grechenig, “Automated security test approach for sip-based voip softphones,” in *The Second International Conference on Advances in System Testing and Validation Lifecycle, August 2010, Nice, France*. IEEE Computer Society Press, Aug. 2010.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “RFC 3261: SIP - Session Initiation Protocol.”

[3] B. P. Miller, L. Fredriksen, and B. So, “An empirical study of the reliability of unix utilities,” *Commun. ACM*, vol. 33, no. 12, pp. 32–44, 1990.

[4] J. E. Forrester and B. P. Miller, “An empirical study of the robustness of windows nt applications using random testing,” in *WSS’00: Proceedings of the 4th conference on USENIX Windows Systems Symposium*. Berkeley, CA, USA: USENIX Association, 2000, pp. 6–6.

[5] P. Godefroid, A. Kiezun, and M. Y. Levin, "Grammar-based whitebox fuzzing," in *PLDI '08: Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation*. New York, NY, USA: ACM, 2008, pp. 206–215.

[6] V. Ganesh, T. Leek, and M. Rinard, "Taint-based directed whitebox fuzzing," in *IEEE 31st International Conference on Software Engineering, 2009. ICSE 2009.*, May 2009, pp. 474–484.

[7] C. Schanes, "fuzzolution fuzzer framework," 2011. [Online]. Available: http://security.inso.tuwien.ac.at/esse-projects/fuzzolution/

[8] H. H. Thompson, "Why security testing is hard," *IEEE Security & Privacy Magazine*, vol. 1, no. 4, pp. 83–86, 2003.

[9] E. Y. Chen and M. Itoh, "Scalable detection of SIP fuzzing attacks," in *Second International Conference on Emerging Security Information, Systems and Technologies, 2008. SECURWARE '08.*, Aug. 2008, pp. 114–119.

[10] P. Oehlert, "Violating assumptions with fuzzing," *Security & Privacy, IEEE*, vol. 3, no. 2, pp. 58–62, Mar./Apr. 2005.

[11] T. Zourzouvillys and E. Rescorla, "An introduction to standards-based voip: Sip, rtp, and friends," *Internet Computing, IEEE*, vol. 14, no. 2, pp. 69 –73, 2010.

[12] D. Endler and M. Collier, *Hacking Exposed VoIP: Voice Over IP Security Secrets & Solutions*. New York, NY, USA: McGraw-Hill, Inc., 2007.

[13] A. D. Keromytis, "Voice-over-ip security: Research and practice," *Security Privacy, IEEE*, vol. 8, no. 2, pp. 76 –78, 2010.

[14] W. Werapun, A. A. El Kalam, B. Paillassa, and J. Fasson, "Solution analysis for sip security threats," in *Multimedia Computing and Systems, 2009. ICMCS '09. International Conference on*, 2009, pp. 174 –180.

[15] C. Wieser, M. Laakso, and H. Schulzrinne, "Security testing of SIP implementations," 2003.

[16] H. J. Abdelnur, R. State, and O. Festor, "Kif: a stateful sip fuzzer," in *IPTComm '07: Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*. New York, NY, USA: ACM, 2007, pp. 47–56.

[17] U. Oulu, "PROTOS: Security testing of protcol implementations." [Online]. Available: https://www.ee.oulu.fi/research/ouspg/Protos,2010-02-14

[18] C. Wieser, M. Laakso, and H. Schulzrinne, "Sip robustness testing for large-scale use," in *SOQUA/TECOS*, 2004, pp. 165–178.

[19] D. Aitel, "The advantages of block-based protocol analysis for security testing," Tech. Rep., 2002.

[20] G. Banks, M. Cova, V. Felmetsger, K. Almeroth, R. Kemmerer, and G. Vigna, "SNOOZE: Toward a stateful network protocol fuzzer," pp. 343 – 358, 2006.

[21] H. J. Abdelnur, R. State, and O. Festor, "Advanced fuzzing in the VoIP space," *Journal in Computer Virology*, vol. 6, no. 1, pp. 57–64, 2010.

[22] T. Alrahem, A. Chen, N. DiGiussepe, J. Gee, S.-P. Hsiao, S. Mattox, T. Park, A. Tam, and I. G. Harris, "INTERSTATE: A stateful protocol fuzzer for SIP," 2007.

[23] I. G. Harris, T. Alrahem, A. Chen, N. DiGiussepe, J. Gee, S.-P. Hsiao, S. Mattox, T. Park, S. Selvaraj, A. Tam, and M. Carlsson, "Security testing of session initiation protocol implementations," *ISeCure, The ISC International Journal of Information Security*, vol. 1, no. 2, pp. 91–103, 2009.

[24] L. Feng and S. Zhuang, "Action-driven automation test framework for graphical user interface (GUI) software testing," *Autotestcon, 2007 IEEE*, pp. 22 –27, sept. 2007.

[25] P. Godefroid, "Random testing for security: blackbox vs. whitebox fuzzing," in *RT '07: Proceedings of the 2nd international workshop on Random testing*. New York, NY, USA: ACM, 2007, pp. 1–1.

[26] J. Neystadt, "Automated penetration testing with whitebox fuzzing," Feb. 2008. [Online]. Available: \url{http://msdn.microsoft.com/en-us/library/cc162782.aspx}

[27] Z. Xiaochun, Z. Bo, L. Juefeng, and G. Qiu, "A test automation solution on GUI functional test," *6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008.*, pp. 1413 –1418, july 2008.

[28] J. Bo, L. Xiang, and G. Xiaopeng, "Mobiletest: A tool supporting automatic black box test for software on smart mobile devices," in *Proceedings of the Second International Workshop on Automation of Software Test*, ser. AST '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 8–. [Online]. Available: http://dx.doi.org/10.1109/AST.2007.9

[29] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, and K. Summers, "Session Initiation Protocol (SIP) Basic Call Flow Examples," 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3665.txt

[30] M. Ronniger, F. Fankhauser, C. Schanes, and T. Grechenig, "A robust and flexible test environment for voip security tests," in *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, Nov. 2010, pp. 1–6.

[31] V. Lebedev, "Qutecom," website, 2011. [Online]. Available: http://www.qutecom.org/

[32] E. Ivov, "Sip communicator," website, 2011. [Online]. Available: http://www.sip-communicator.org/

Genomics-based Security Protocols: From Plaintext to Cipherprotein

Harry Shaw
Microwave and Communications Systems Branch
NASA/Goddard Space Flight Center
Greenbelt, MD, USA
harry.c.shaw@nasa.gov

Sayed Hussein, Hermann Helgert
Department of Electrical and Computer Engineering
George Washington University
Washington, DC, USA
drsay@gwu.edu, hhelgert@gwu.edu

***Abstract*—The evolving nature of the internet will require continual advances in authentication and confidentiality protocols. Nature provides some clues as to how this can be accomplished in a distributed manner through molecular biology. Cryptography and molecular biology share certain aspects and operations that allow for a set of unified principles to be applied to problems in either venue. A concept for developing security protocols that can be instantiated at the genomics level is presented. A DNA (Deoxyribonucleic acid) inspired hash code system is presented that utilizes concepts from molecular biology. It is a keyed-Hash Message Authentication Code (HMAC) capable of being used in secure mobile ad hoc networks. It is targeted for applications without an available public key infrastructure. Mechanics of creating the HMAC are presented as well as a prototype HMAC protocol architecture. Security concepts related to the implementation differences between electronic domain security and genomics domain security are discussed. This paper demonstrates a practical path to a composable, standardized biological internet security protocol that encompasses biological and computing domains.**

Keywords-HMAC; keyed Hash Message Authentication Code; Cryptography; DNA; PKI; public key infrastructure; MANET; cipherprotein; epigenetics; security architecture

I. INTRODUCTION

The ability to authenticate the identity of participants in a network is critical to network security. Bimolecular systems of gene expression "authenticate" themselves through various means such as transcription factors and promoter sequences. They have means of retaining "confidentiality" of the meaning of genome sequences through processes such as control of protein expression. These actions occur independently of a centralized control mechanism. The overall goal of the research is to develop practical systems of authentication and confidentiality such that independence of authentication and confidentiality can occur without a centralized third party system.

Genes are capable of expressing a wide range of products such as proteins based upon an alphabet of only four symbols. This research implements a keyed-HMAC system using a DNA-based code and certain principles from molecular biology. The system will permit Mobile Ad hoc Networks (MANET) to distinguish trusted peers, yet tolerate the ingress and egress of nodes on an unscheduled, unpredictable basis. The system allows for authentication without a Public Key Infrastructure (PKI), X.509 certificates, RSA and nonce exchanges, etc. It also provides for a biological authentication capability.

This paper will move between the electronic and genomics contexts when discussing the protocols and their potential instantiation. This scheme can be used to create encrypted forms of gene expression that express a unique, confidential pattern of gene expression and protein synthesis. The ciphertext code carries the promoters (and reporters and regulators) necessary to control the expression of genes in the encrypted chromosomes to produce cipherproteins. Unique encrypted cellular structures can be created that can be tied to the electronic hash code to create biological authentication and confidentiality schemes.

This paper is organized as follows:

- Background information on the state of the art and a description of the elements of the prototype genomic HMAC architecture
- A description of the DNA code encryption process, genome selection and properties
- The elements of the prototype protocol architecture and its concept of operations
- A short plaintext to ciphertext encryption example.
- Description of the principles of gene expression and transcriptional control to develop protocols for information security. These protocols would operate in both the electronic and genomic domains.

II. BACKGROUND OF DNA CRYPTOGRAPHY

The use of DNA as a cryptographic medium is not new. DNA encryption systems are one of the paths taken in the field of molecular computing. Systems using DNA as a one-time code pad in a steganographic approach have been described [1], [2]. An image compression – encryption system using a DNA-based alphabet [3] was demonstrated including a genetic algorithm based compression scheme. Schemes utilizing DNA encryption utilizing dummy sequences of DNA have been published [4]. The steganographic approach is highly desirable because DNA provides a natural template for the hidden message approach [5]. It also appears in applications such as DNA watermarks [6] and specifically DNA watermarks to identify genetically

modified organisms utilizing the DNA-Crypt algorithm. [7]. This algorithm permits a user to insert encrypted data into a genome of choice.

The research described herein is not just about inserting encrypted sequences into genomes. It will insert messages that can control gene expression through a variety of mechanisms. It is also focused on a broader goal of extending biological mechanisms that control gene expression into a domain that includes network authentication.

III. ELEMENTS OF THE GENOMICS HMAC ARCHITECTURE

Plaintext is mapped into a reduced representation consisting of an alphabet of q letters, where q = 4 for a genomic alphabet such as DNA or Ribonucleic acid (RNA), q = 20 for proteomic alphabet, or other values when representing other functions in molecular biology, e.g., histone code. The actual HMAC requires additional base representations beyond the four DNA bases, but the minimum requirement is shown in (1) and (2) [8]. B is the set of DNA bases A, T, C and G, which represent the molecules adenine, thymine, guanine and cytosine and represent the entire alphabet of the genomic hash code. DNA bases have the property that the only permitted pairs are Watson-Crick matches (A-T), (C-G), thus, the binary representations of B and B' sets are complimentary such that a r-bit length sequence of B_q and B'_q maintain the identity property shown in (3). Assignment of letter to DNA base sequences is performed. Letters with greater frequency can be assigned shorter DNA sequences to reduce the code size.

A. Lexicographic and DNA representation of plaintext

Plaintext words, P are converted into a numerical form suitable for subsequent coding into the cryptographic alphabet of the required code. Plaintext words are coded such that a lexicographic order is maintained between words, i.e., the numerical forms may take either integer or floating point representations. F is a function that converts the plaintext to lexicographic numerical form. D represents the numerical form of the dictionary (lexicographically ordered set) such that $D_{1,..n}$ represents the set of all words. The subset of $D_{1,..i}$ represents the subset of words in the plaintext message. The function U assigns the DNA base sequence corresponding to the D_i as shown in (4), (5) and (6). L is the plaintext message coded into the DNA alphabet found in sets B and B'.

B. Sentence-message order coding

A system of linear equations codes the lexicographic position of each word relative to the sentence position of each word. This complicates detection of words based upon frequency analysis. Multiple appearances of the same word are uniquely coded. As a minimum requirement, if there are i DNA representations in the message, and n represents a numerical sequence related to the number of DNA representations in the message (the simplest case being i = 1, 2, 3, …, n), then the system of linear equations shown in (7) provide the solutions for sentence-message order coding.

$$B_q = \{A, T, C, G\} \tag{1}$$

$$B'_q = \{T, A, G, C\} \tag{2}$$

$$1 = B_q^r \oplus B_q^{r'} \qquad \forall r = 1, ..., q \tag{3}$$

Equations 1 and 2 define the sets containing the DNA bases that comprise the alphabet for the HMAC code. Equation 3 defines the complimentary relationship required for the binary representations of the members of that space. For example: the XOR product of the r^{th} bit of A and T is a one as is true for T and A, C and G, G and C.

Equation 4 defines each word in the message, P_i as a member of a set of all words in a lexicographically ordered dictionary. Equations 5 and 6 show the operation of the function that assigns a DNA sequence using the members of the set of DNA bases to a coding of concatenated sequences labeled L and L'. L and L' maintain the same complimentary relationship that is a property of the individual DNA bases in the sets B_q and B'_q.

This yields a series of coefficients $x_1, x_2, \ldots, x_i$ that are concatenated as shown in (8). The binary representation of each coefficient undergoes bit expansions such that only B_q or B'_q codes are represented in the bit stream created by (8). X represents the relationship between lexicographic coding of the words and their position in the message.

$$D_i = F(P_i) \quad \ni \quad D_i < D_{i+1} \quad \forall \quad i < n \tag{4}$$

$$L = U(D_1, B_q) \| U(D_2, B_q) \| ... \| U(D_i, B_q) \tag{5}$$

$$L' = U(D_1, B'_q) \| U(D_2, B'_q) \| ... \| U(D_i, B'_q) \tag{6}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_i \end{bmatrix} = \begin{bmatrix} D_1 & D_2 & \ldots & D_i \\ D_i & D_1 & \ldots & D_{i-1} \\ \multicolumn{4}{c}{\cdots\cdots\cdots\cdots} \\ D_2 & D_3 & \ldots & D_1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ .. \\ r_i \end{bmatrix} \tag{7}$$

$$X = x_1 \| x_2 \| ... \| x_i \tag{8}$$

$$M = L \oplus X \tag{9}$$

C. Message coding

DNA coding on the message is completed by XOR and bit expansions to maintain the DNA base coding in the binary sequence in the operation shown in (9). M is the plaintext message coded into the DNA alphabet and coded again with the sentence-message coefficients. This sequence will be subjected to encryption.

The set of linear equations in equation 7 provide the process of sentence-message order coding using the r^{th}

position in the message to code each word of the message. The resulting coefficients are concatenated and XOR'd with the coded plaintext message to produce the ciphertext message.

IV. ENCRYPTION PROCESS

Approximately 800 genomes have been sequenced [9]. The human genome alone has approximately 3.2 million base pairs. The sets of genomes provide for the possibility of "security by obscurity". Additionally, there is an infinite number of ways to use genome sequences as cryptographic keys. However, genomes have high degrees of redundancy and sequence conservation across species. Consequently, sections of genomes used as keys should be treated as one-time pads. The first step is to select a genome and a sequence from that genome and encode it with the binary representations of B_q and B'_q.

DNA consists of two complimentary sequences, referred to as the sense and antisense strands as shown in Fig. 1 [10]. A DNA sequence has a start point called the five-prime end (5') and an endpoint called the three-prime (3'). In biochemistry, the 5' and 3' designations refer to orientation of each strand necessary for proper replication and transcription. The complements are bonded to each other base by base to create base pairs. The antisense strand is oriented in the 3' to the 5' direction, relative to the sense strand. For a DNA encryption key, both sense and antisense strands can be encoded and utilized. Figs. 2 and 3 demonstrate two ways of implementing the chromosome encryption key in the HMAC scheme. Fig. 2 represents the simplest scheme, in which successive bases from the key and message are XOR'd and a single ciphertext message is produced. Encryption proceeds in the 5' to 3' direction using the sense strand. Fig. 3 represents a more complex scheme, in which both sense and antisense bases from key and message are XOR'd. Encryption proceeds in the 5' to 3' direction in both strands.

A. Mismatches and Annealing

The encryption process generates base pair mismatches that do not conform to the A-T, C-G pairing rule. These mismatches are central to creating a one-way hash code. Subsequent to the encryption step, the mismatches are resolved through an annealing process that results in an irreversible transformation of the encryption sequence not directly traceable to the original ciphertext.

V. PROTOTYPE DNA-BASED, KEYED HMAC SYSTEM

Assume a network such as the one shown in Fig. 4. Jack, Jill, JoAnn and Lisa wish to form a secure MANET. In the same wireless transceiver space can be found X and Y whose intentions are unknown, but are capable of sending and receiving messages. Jack, Jill, JoAnn and Lisa possess all of the required authentication tools:

- A common genome, C, to use as an HMAC key.
- A pre-shared secret, pss, unique to each party.
- The DNA-based HMAC algorithm.

Consider two authentication scenarios. In the first scenario Jack, Jill, JoAnn and Lisa send and receive cleartext messages using the DNA-based HMAC authentication. If the receiver is not the intended destination, the receiver rebroadcasts the message with their hash and the process continues until the message reaches the intended receiver or until a message time-out period elapses. X and Y also receive the cleartext messages and hash codes. X and Y may possess the algorithm. However, if X and Y wish to substitute a new message with a valid hash code, or forward the message and have it accepted by the network members, they have to create a valid hash code and checksum, which requires knowledge of the chromosome sequence and valid pre-shared secrets known to the other MANET nodes. The MANET members change their pre-shared secrets on a pre-established basis to thwart a brute force attack to derive the pre-shared secret from the hash code.

In the second scenario, Jack, Jill, JoAnn and Lisa wish to establish a trust relationship before exchanging sensitive information across a MANET. In this case, the participants utilize a confidentiality (encryption) protocol for the messages and establish a chain of custody using keyed HMAC authentication. A hash chain of hash codes is established such that each recipient can determine the origin and subsequent hops of the message. In this case, X and Y cannot read the plaintext and the hash code transcript may be encrypted and compressed with the ciphertext.

A. Genomic hash code properties

Table 1 summarizes the properties of the prototype hash code against the requirements for an ideal hash code [11]. Fig. 5 provides a flow chart of the genomic hash coding process.

B. Initialize and Perform Lexicographic and DNA assignments

The plain text message is read and parsed into 3-word blocks (3WB). Take each word in the string, assign it a lexicographic value of x.yyyy….y where x = 1,....,26 corresponding to the first letter of the word and subsequent letters are assigned to each successive decimal place until the entire word is coded in a rational number. Assign a DNA letter code to each letter. Most common English alpha characters use 2-letter codes, the rest use a 3-letter code as shown in Table 2. The column labeled 'α' is the English alphabetic character adjacent to its DNA code equivalent. As an example, the short phrase 'jump out windows' is shown in its lexicographic and DNA assigned forms in Table 3.

C. Binary representation of the DNA bases

The four DNA bases (A, T, C, G) are represented by binary sequences (0011, 1100, 1001, 0110). The remaining 12 four-bit sequences code for transitional base sequences that are used to anneal mismatches in the encryption process as shown in Table 4. The 'Key' column represents the base in the chromosome encryption key. The 'M' column represents the corresponding base in the DNA coded message.

The 'Result' column represents the results of encrypting the key onto the message. The 'Anneal' column represents the final ciphertext base. In an operational system, all codes would be significantly lengthened to thwart brute force attacks.

TABLE 1. GENOMIC HASH CODE PROPERTIES.

Property	Compliance
Produces a fixed length output.	2560 bits
Can be applied to a block of data of any length	Yes.
H(x) is relatively easy to compute for any message x.	Yes. 12 step process for hash code.
One-way property. For any h, it is computationally infeasible to find H(x)=h	To be determined
Weak collision resistance. For a set of x_i messages, with $y \neq x_i$ for all i, no $H(y)=H(x_i)$ for all i.	Yes.
Strong collision resistance. For any x, with $y \neq x$, no H(y)=H(x)	No. Messages $\leq$ 512 bits require padding

TABLE 2. SAMPLE OF ALPHA TO DNA CONVERSION CODES.

α	DNA	α	DNA	α	DNA	α	DNA
0	CGG	G	TT	N	TG	U	CT
A	GC	H	AC	O	AG	V	CTG
B	TGT	I	AA	P	GA	W	CAC
C	TC	J	AAG	Q	CCT	X	GTA
D	GT	K	ACT	R	CC	Y	GTT
E	TA	L	AT	S	GG	Z	TAG

TABLE 3. PLAINTEXT TO LEXICOGRAPHIC ORDER AND DNA LETTER CODES.

#	Conversion		
	Plain text	*Lexicographic Conversion*	*DNA Conversion*
1	jump	10.211316	AAGCTCGGA
2	out	15.2120	AGCTCA
3	windows	23.9144152319	CACAATGGTAGC ACGG

TABLE 4. ENCRYPTION AND ANNEALING TABLE.

Key	M	Result	Anneal	Key	M	Result	Anneal
A	T	T	G	C	G	G	A
A	A	gA	C	C	A	aA	C
A	C	gC	T	C	C	aC	G
A	G	gG	A	C	T	aT	T
T	A	A	T	G	C	C	C
T	G	cC	G	G	A	tA	G
T	C	cG	A	G	G	tG	A
T	T	cT	C	G	T	tT	T

D. Encryption, Mismatches and Annealing

Fig. 5 also provides a short example of the encryption and annealing process. Each base in the chromosome is XOR'd against the corresponding base in the message. If the base in the message is the complement of the base in the chromosome, the base in the message is copied to the encrypted output string and then altered to a new base in the annealed output string If the base in the message is not the complement of the base in the chromosome, a transitional base, whose value depends upon the mismatch is written to the encrypted output string. The 5' base always determines the change in the other strand; consequently, a 5' G mismatch always codes for a 3' transitional base. This feature allows tracking of point mutations and provides a future expansion capability for mutations. The annealing process also alters the encrypted result by transforming the positions that are not mismatches.

E. Cryptographic Genome

Mycoplasma genitalium G37 (National Center for Biotechnology Information accession number NC000908.2) is the bacterial genome used as an encryption key in the prototype system. There are a number of characteristics of *M. genitalium* that make it a good candidate as an encryption key base. It is small (it may be the smallest, self-replicating genome). It has 580,070 base pairs with 470 predicted coding regions. *M. genitalium* has a low G+C content of 34% [12]. A random, uniform distribution of basepair content would provide for 50% G-C pairs and 50% A-T pairs. This feature provides some testability advantages. The genome contains 470 predicted protein coding regions, which is a manageable number of potential cipherproteins. Knowledge of the genome coding characteristics is important in selecting and utilizing genomes as cryptographic keys. Approximately 62,000 base pairs are being utilized from the *M. genitalium* genome for the prototype HMAC.

F. Protocol for Message Authentication.

The process is as follows:

- Encode the plaintext message into DNA code (Pre-sense message) 3 words at a time (3 word blocks – 3WB)
- Encrypt with pre-shared secret chromosome key and generate sense and antisense strands.
- Different chromosome segments are used to encrypt each 3WB for increased key confidentiality.
- Combine sense and antisense strands to create a checksum (S).
- Anneal the sense strand (Sender) or the antisense strand (Receiver) removing the transitional bases in the 3WBs.

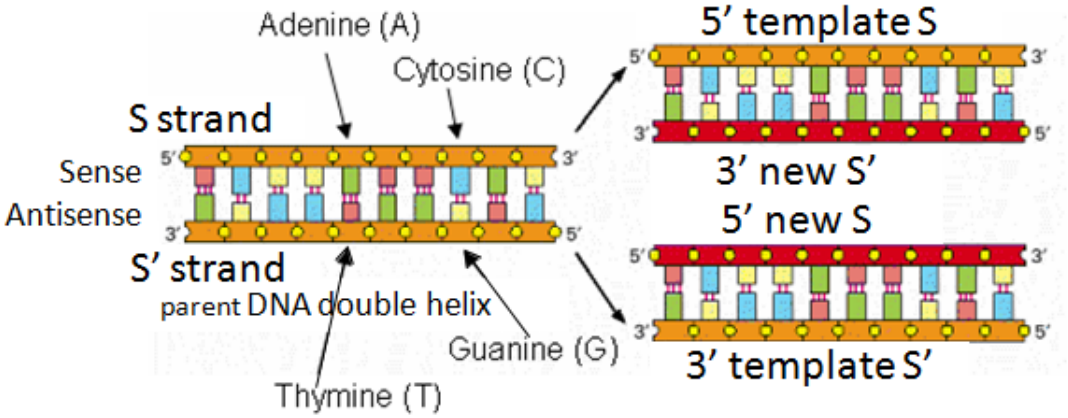


Figure 1. Strand Sequence specification in DNA. A binds with T, C binds with G.

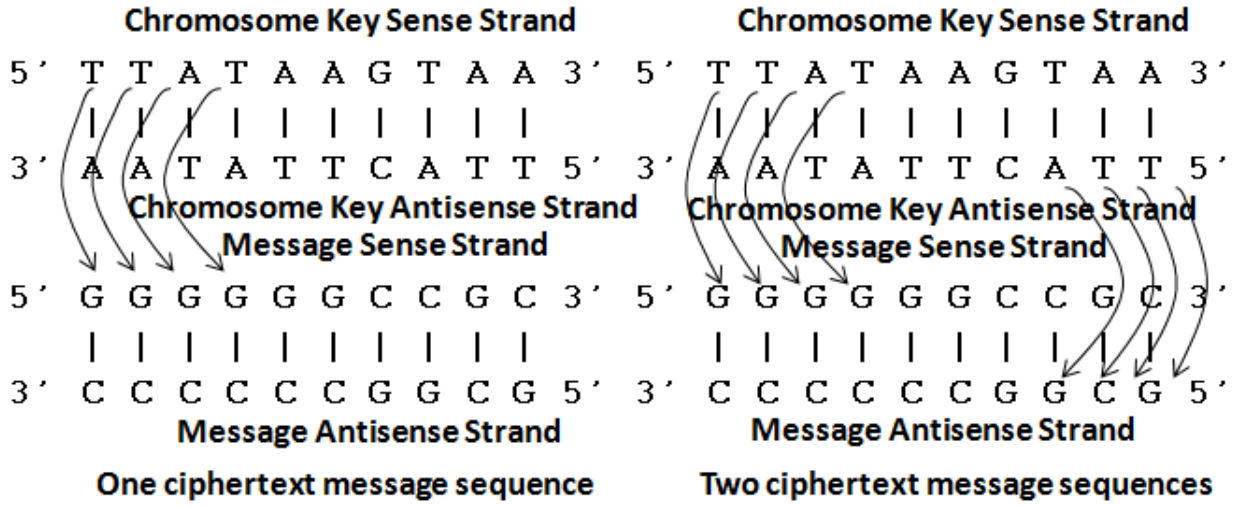


Figure 2. Single strand chromosome encryption utilizing yielding a single ciphertext message sequence.

Figure 3. Dual strand chromosome encryption yielding two ciphertext message sequences

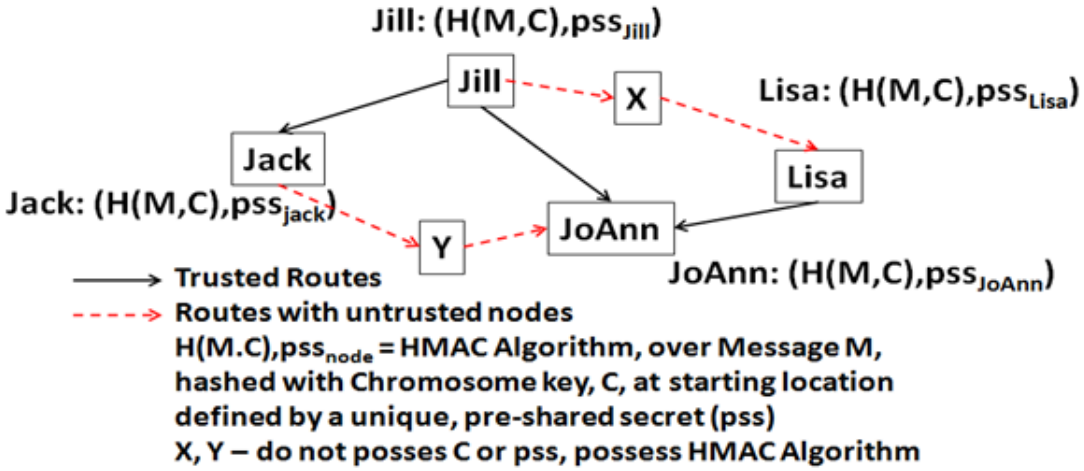


Figure 4. Mobile Ad-hoc Network with trusted and untrusted nodes and routes.

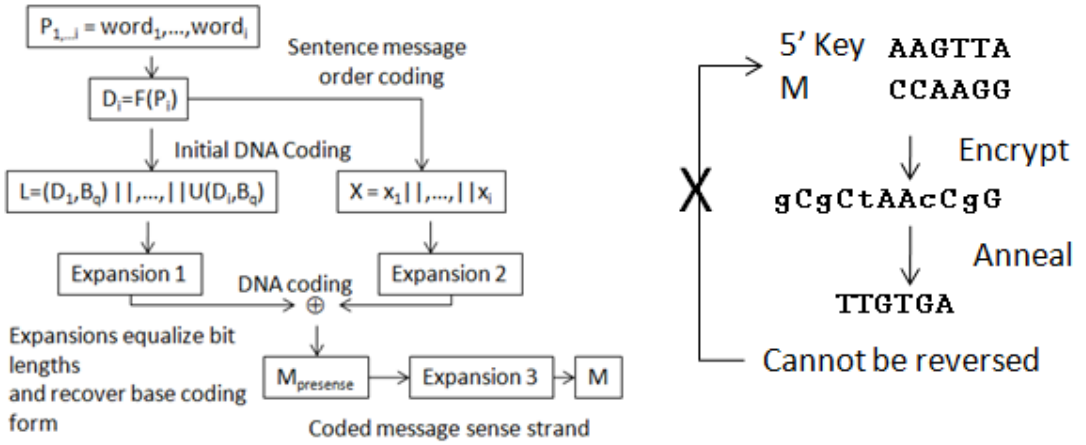


Figure 5. Plaintext coding process, Encryption and Annealing process

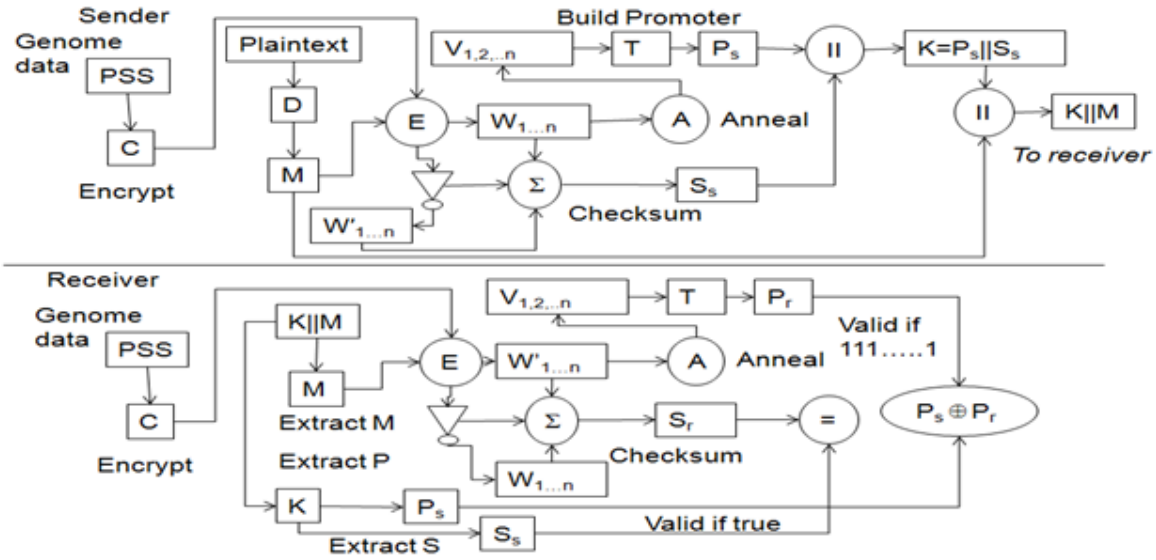


Figure 6. Sender and Receiver Protocol.

DNA Sense strand coded message
AAGCTCGGAAGCTCACACAATGGTAGCACGG

***M genitalium* key**
TTTAGTTATAAGTTATTATTTAGTTAATAAGTTATT
ATT..... GTTATTATTTAGT

checksum = 1871221
'----------Sender copy of hash----------
Sender Hash =
AAAAAAGTATTCTTTGTCAGTGTTTGTGCGTTTCAACCCCTC
AATTAGATTTATAGAGTCCTTC
Plain Text =
jump out windows /
Message Builder 4 - for DNA Hash Code System ended at:
5/21/2009 1:24:47 PM

'----------Receiver copy of hash----------
Sender Hash =
AAAAAAGTATTCTTTGTCAGTGTTTGTGCGTTTCAACCCCTC
AATTAGATTTATAGAGTCCTTC
Receiver Hash =
TTTTTTCATAAGAAACAGTCACAAACACGCAAAGTTGGGGA
GTTAATCTAAATATCTCAGGAAG
Sender Checksum = 1871221
Receiver Checksum = 1871221
Plain Text = jump out windows /
Checksum matched
hash code matched
User and Message Authenticated!
Message Reader 4 for DNA Hash Code System ended at:
5/21/2009 1:32:09 PM

Figure 7. Sample output of sender and receiver performing authentication on the cleartext phrase 'jump out windows'.

- Concatenate the first 64 DNA bases from the first nine 3WBs to create the Promoter (P).
- Append the checksum to the Promoter. The Promoter || checksum is the Hash Code, K (2560 bits long). The sender and receiver processes are summarized in Fig. 6.

The receiver extracts the Promoter and checksum from the message. The hash code computed at the receiver must have the complement of the Promoter sequence and an exact match of the checksum. Sender and receiver must have the pre-shared secret of the genome, and the location of the first base of the sequence. A sample of the output for the test message 'jump out windows' is shown in Fig. 7. The hash code has been truncated for test and presentation purposes.

G. Short Message Performance

A critical factor in determining the goodness of a hash code is the ability to satisfy criteria four and five from Table 1. A hash code algorithm should not produce identical hash code outputs for two or more different messages. Performance of short messages was evaluated for soft and hard collision resistance. The number of MAC verifications, R, required to perform a forgery attack on a m-bit MAC by brute-force verifications [13] is shown in equation 10:

$$R = 2^{m-1} + (2^{m-1} - 1)/2^m \approx 2^{m-1} \quad (10)$$

The variable R is an approximate upper bound to the brute-force verification limit. Short messages were repeatedly hashed using over different cryptographic sequences to look for collisions. The process is shown in Fig. 8. Table 5 summarizes the results of those tests.

The single letter message exhibited 403 checksum collisions and 466 hash code collisions. Chromosomes have a high degree of redundancy and repetition; therefore short messages will require padding to eliminate hash code collisions. These statistics utilize different transcripts on the same message to identify potential collisions. These statistics should be indicative of the potential for multiple messages to produce the same hash code from a single transcript. For secure authentication purposes, this code must be implemented with higher level protocols that would block a brute force attack and not reuse genome sequences for authentication. It must also move the starting point in the genome to widely separated start positions to prevent an attacker from guessing the encryption sequence.

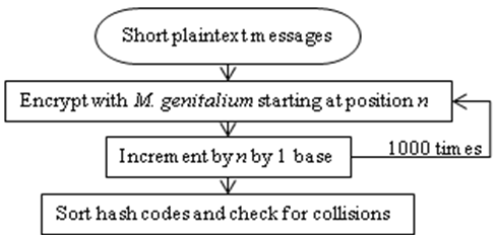


Figure 8. Collision resistance tests for short messages.

TABLE 5. SAMPLE OF HASH CODE COLLISIONS

Plain Text	Msg Length	Hash Code Length	Total Hash Code Collisions	Total C/S Collisions	R
z	1	22	466	403	2097152.5
ly	2	30	255	214	536870912.5
cat	3	36	136	109	34359738369
vent	5	64	0	0	9.22337E+18
aeiou	6	64	0	0	9.22337E+18
jump out windows	16	64	0	0	9.22337E+18
jump out windows jump out windows jump out windows jump out	59	256	0	0	5.7896E+76
the 123 of my fields are very large please require all personnel to take their equipment with them for the work to be performed in 365777 small increments it will be good to get practice on these tasks	201	576	0	0	1.2367E+173

A hash code must be secure against the possibility that the cryptographic key, in this case the original genome sequence cannot be recovered from the hash code. Fig. 9 represents a small MANET example for developing trust metrics.

Assume Jack is broadcasting forward requests to establish a link with Lisa and Lisa is broadcasting return route requests to Jack to establish a return link. Jill is relaying route requests in both directions. Felix wishes to join the MANET. Each node is capable of dynamically appearing and disappearing from the network at will via application of a dynamic source routing protocol. Each node can also take the role untrusted/unknown trust or trusted depending upon the situation. Source and Destination must determine the trustability of a potential route through some quantitative means. In this case successful forward and return route requests (FREQ, RREQ) and route delays are used to create the trust metrics. The sources and destinations can set the minimum level of trust for routes via a dynamic fitness algorithm.

To establish Felix as a trusted member, he relays forward REQs from Jack destined for Lisa and return REQs from Lisa destined for Jack with his DNA HMAC authentication attached. JoAnn, does not respond to route requests and those requests time-out.

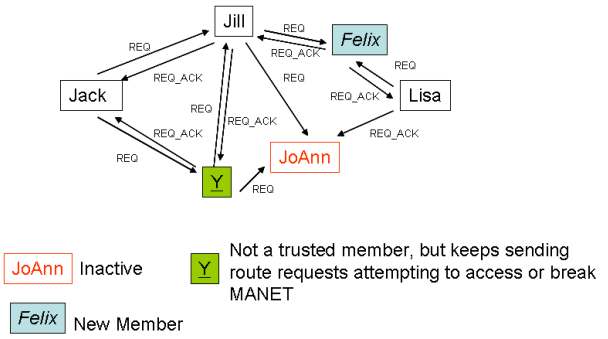


Figure 9. MANET route establishment at a slice in time.

Y is a malfeasor attempting to breach the network by sending route requests with counterfeit DNA HMAC authentication and analyzing received DNA HMACs for vulnerabilities. Assume that when Y sends a counterfeit route request, genuine nodes respond with negative acknowledgement attached to a genuine authentication code. The questions to be answered are:

- Can Y establish a counterfeit authentication code (hash + checksum) for the current session (however a session is defined)?
- Can Y utilize the stolen information to recover information that might be useful for a future network breach?

If Y can recover the original cryptographic sequence, or determine the genome and genome location a cryptographic key was taken from, Y may be able to forge a valid hash code. This could be problematic for a cryptographic sequence due to the high degree of redundancy in the all genomes. For this application, the hash code must be evaluated against the cryptographic key to ensure it has the proper characteristics of diffusion and confusion.

VI. MUTATION EFFECTS, FITNESS, DIFFUSION AND CONFUSION

Life is intolerant of a high mutation rate in its genetic code. Ribonucleic acid (RNA) viruses have the highest mutation rate of any living species, 10^{-3} to 10^{-5} errors/nucleotide and replication cycle [14]. The human DNA mutation rate has been approximated to be on the order of 10^{-8} errors/nucleotide and generation [15]. Injection of mutations into DNA encrypted messages is an approach to improving the encryption process. Because of the dynamic, evolutionary nature of this approach, potential intruders must continually intercept decoding instructions between source and destination. Missing one generation of genome decryption information seriously corrupts the analysis process. Missing multiple generations eventually renders previous decryption analyses useless.

In evolutionary biology, fitness is a characteristic that relates to the number of offspring produced from a given genome. From a population genetics point of a view the relative fitness of the mutant depends upon the number of descendants per wild-type descendant [16]. In evolutionary computing, a fitness algorithm determines whether candidate solutions, in this case encrypted messages, are sufficiently encrypted to be transmitted. This DNA encryption method uses evolutionary computing principles of fitness algorithms to determine, which encrypted mutants should be selected as the final encrypted ciphertext. Two parameters, Confusion and Diffusion are being used as the basis of the fitness criteria. Diffusion and Confusion are fundamental characteristics of ciphers. Shannon [17] describes them as:

1) Diffusion: any redundancy or patterns in the plaintext message are dissipated into the long range statistics of the ciphertext message.

2) Confusion: make complex the relationship between the plaintext and ciphertext. A simple substitution cipher would provide very little confusion to a code breaker.

The challenge is to create a set of FREQ and RREQ messages that hash into codes with a high degree of diffusion and confusion. One strategy for attacking the authentication message is to generate long strings of zeros and identify the correct code for the non-zero positions. If a message generates long strings of zeros it is particularly vulnerable to a key recovery attack because the attacker can reduce the number of bit matches required by the length of zero bit blocks. Table 7 summarizes test results of 1000 trials on messages consisting of zeroes and spaces against the genome. No collisions were identified. The hash code will be tested against all other single character strings to identify patterns. A sample hash code of a string of 192 zeros is shown below in Table 6.

TABLE 6. SAMPLE HASH CODE STRING OF 192 ZEROS

Checksum	DNA Hash Code
10437404	AATTCTAAGTTCCCGCCCGTCGGTCCGCCGCCC GTCCGGTCCGCCGCCCGTCCCGGTCCGCCGCA ATCTCAATTCTCGCCCGTCGGTCCGCCGCCCGT CCGGTCCGCCGCCCGTCCCGGTCCGCCGCCAA CTCCAATCTTGCCCGTCGGTCCGCCGCCCGTCC GGTCCGCCGCCCGTCCCGGTCCGCCGCCCAAT CCGAACTTCCCCGTCGGTCCGCCGCCCGTCCG GTCCGCCGCCCGTCCCGGTCCGCCGCCCGAAC CGTAATTCTCCGTCGGTCCGCCGCCCGTCCGGT CCGCCGCCCGTCCCGGTCCGCCGCCCGTAACG TTAATCTTCGTCGGTCCGCCGCCCGTCCGGTCC GCCGCCCGTCCCGGTCCGCCGCCCGTCAAGTT CAACTTTAATCCGAACTTCAATCGTAACGTTA ATCTTTCGTTTAAGTTCAACTTTAATTAATTCT AATTTCAACCGTAATTCTAACGTTAAGTTCAAC TTTCGTTTCAATTCTAATTTCAATC

TABLE 7. TEST RESULTS ON REDUNDANT STRINGS OF ZEROES MESSAGES

Length of '0's in Plain Text	Number of Collisions after 1000 trials
64	0
96	0
192	0

Next the hash codes were compared to the original cryptographic keys to evaluate diffusion and confusion. Table 8 displays four mutation samples from 50 combinations of hash codes on the message 'jump out windows' with encryption keys from the genome. The process was run on 1000 message combinations at a time. Mutants 4 and 25, for example would be particularly poor fits due to the number of consecutive matches between the hash code and encryption key. Mutant 10 has only one match of two consecutive bases and a fewer than ¼ of the bases are identical between the hash code and key. Each position in the hash code has 1 of 4 chance of randomly matching the same location in the encryption key. The confusion metric counts the number of 2-base, 3-base, 4-base and 5-base consecutive matches between the hash code and the key. Each combination actually represents a mutant message, which can be further evaluated via a genetic algorithm. One of the major advantages of this system of a conventional encryption system is the ability to provide a set of encrypted outputs from, which the most fit (best) member can be selected.

TABLE 8. SAMPLE MUTANT ENCRYPTIONS FOR HASH CODES AND DNA ENCRYPTION KEY FOR MESSAGE 'JUMP OUT WINDOWS'

ID	64 Base Pair Hash Code	Cryptographic key
Mutant 4	AAAAAATGATGGTCCGC CAGTGCTCCGGCTCTCCA ATGCCTGAATCAGATGG AGAGATTCTGGC	TAAGTTATTATTTA GTAAGTTATTATTT AGTTAAGTTATTAT TTAGTTTAAGTTATT ATTTAGT
Mutant 10	AAAAAACGATGGCTGGC GATCTCTCCGTTCCCGTA ACTCCTGAAGGATAGCT ATAGATTCCCTC	TTATAAGTTATTATT TAGTAAGTTATTAT TTAGTTAAGTTATT ATTTAGTTTAAGTT ATTATTT
Mutant 23	AAAAAAGGAGGGCGGG CCAGTGCTCCGGCTCTTC AATCGCGTAAGTAGATC CACAGAGTGTCTG	AAGTTATTATTTAG TTAAGTTATTATTTA GTTTAAGTTATTATT TAGTTATAAGTTAT TATTTA
Mutant 25	AAAAAAGGAGGTTTGTG TAGCGTTTGGGCCCTCG AACCGGCGAAGGAGAGG GAGATATCTTCCC	GTTAAGTTATTATTT AGTTTAAGTTATTA TTTAGTTATAAGTT ATTATTTAGTTAAT AAGTTAT

TABLE 9. SAMPLE DIFFUSION AND CONFUSION SCORES FOR HASH CODE FOR MESSAGE 'JUMP OUT WINDOWS'

ID	Diffusion - matching base pair positions	Confusion - consecutive match positions			
		2	*3*	*4*	*5*
Mutant 4	25	9	5	3	2
Mutant 10	11	1	0	0	0
Mutant 50	14	3	0	0	0
Mutant 25	21	5	1	0	0

A. *Intronic sequence padding and potential frameshift mutations can increase cryptographic hardness*

Padding short messages and short words has been previously discussed as a means to decrease collisions and reduce the likelihood of successfully forging messages. Adding padding to the front of messages as well as the end and padding short words makes it more difficult for an attacker to find the start of the coded message sequence. The analogy in molecular biology is the frameshift mutation, in which changing the starting position for a single nucleotide can result in a completely different protein sequence as shown in Fig. 10. The mechanics of DNA transcription in cells relies on a number of properties to identify the nucleotide triplet sequence that actually transcribes to mRNA, which translates to a protein. Some of the mechanics are thermodynamic and biochemical in nature such as DNA folding, binding to transcription factors, and chromatin relaxation in eukaryotes. Some of the mechanics are sequence related. Four types of sequences and mechanisms from molecular biology are directly relevant to this discussion:

1) Start codon: (usually ATG) to specify the transcription start site (three letter sequence that ultimately specifies the first amino acid in the protein to be translated.)

2) Stop codon: (TAA, TGA, TAG) to end transcription.

3) Promoters. The function of promoters is different in prokaryotes and eukaryotes, but as a general statement, the promoter is sequence of nucleotides necessary to locate the transcription starting point. In eukaryotic genes that contain a promoter, the sequence often contains the letters 'TATA' hence the term 'TATA box'.

4) Enhancers. In eukaryotes, a variety of sequences upstream and downstream from the transcription site provide binding sites for transcription factors (proteins) necessary to enhance protein expression.

The transcription (decryption) of DNA uses these sequences as markers for process control. But the sequences can have multiple interpretations. ATG within a gene codes for the amino acid methionine; at the start of a gene it is a start codon. All instances of TATA do not signify a promoter. These ambiguities provide DNA with its own version of adding diffusion and confusion, and the analyst must fully understand the rules and mechanisms of transcription. In fact, research in gene expression starts with unambiguously identifying the actual gene sequence that codes for proteins (in eukaryotes this is called the exon region) from intervening sequences that are untranslated regions that do not code for proteins (intron regions) as shown in Fig. 11 for the human gene *hspB9*, which codes for heat shock protein B9 (Ensembl ENSG00000197723). Referring back to Fig. 10, transcription from a different start site would yield a different outcome, one that is possibly fatal to the organism. Padding creates introns spread throughout the message (exon).

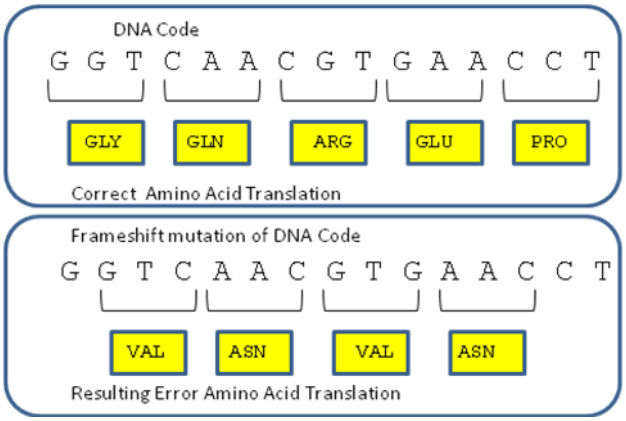


Figure 10. Frameshift Mutations

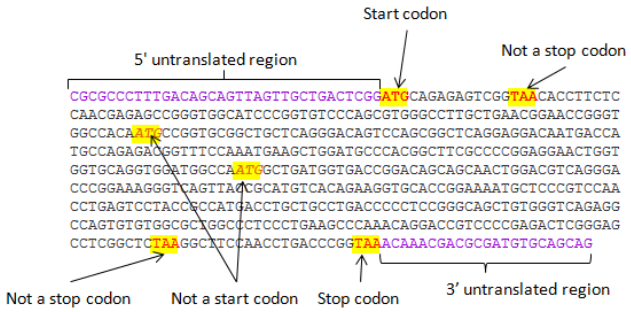


Figure 11. Confusion factors in actual DNA genome

The same confusion and diffusion factors would apply when crafting DNA coded messages for the electronic domain that will be later instantiated into actual genomes. The ciphertext must be capable of meeting the requirements of the cryptographic hardness in the electronic domain while producing a ciphertext that can be reliably integrated into a cellular genome via standard techniques, transcripted into RNA, and translated into the appropriate cipherprotein. Decryption (expression) of the cipherprotein gene occurs in response to specific decryption instructions hidden within the electronic domain ciphertext .

VII. Relationship Between Cryptography and Gene Expression

The following relationships can be observed between the cryptographic treatment of messages and control of gene expression. In the case of gene expression, the message is genomic (DNA or RNA sequence).

- Cryptography transforms messages between two states: plain and encrypted.
- Cryptography uses operations such as circular shifts, bit expansions, bit padding, arithmetic operations to create ciphertext. These operations have analogs in molecular biology, e.g., transposable elements
- Cells transform DNA sequences in genes between two states: Expressed (decrypted) and Silent (encrypted)
- In prokaryotes a simple system involving operators and repressors can be described in terms of encryption and decryption, but prokaryotes have fewer mechanisms available for a rich set of cryptographic protocols. Fig. 12 provides an example from *Escherichia coli* using *lacZ* gene expression.

In this prokaryotic example from *E. coli*, the *lacZ* gene expresses the β-galactosidase enzyme when lactose is present and the simple sugar glucose is absent. β-galactosidase metabolizes lactose into glucose and galactose. It would be inefficient to express the enzyme above a trace level if glucose is present. Fig. 12 provides a cryptographic analogy to the states of the *lacZ* gene under the various conditions of glucose and lactose present, lactose present, and lactose absent. The *lacZ* gene is encrypted when lactose is absent or both lactose and glucose are present. A repressor protein (rep) authenticates (binds) to the encryption site (*lacZ* operator) on the *lacZ* gene with lactose is absent. A catabolite activator protein (CAP) authenticates (binds) to the decryption site (CAP site) allowing RNA polymerase to decrypt (express) the *lacZ* gene when glucose is absent. All of these operations are shown as analogies to elements of cryptographic message traffic in operations shown in Fig. 12. It is possible to write the description of the gene expression sequence in Fig. 12 in terms of a series of messages between a sender and receiver.

Fig. 13 shows the architecture of the DNA HMAC (without all the required control regions) described in detail in this paper and its comparison between gene transcriptional control structures for a typical mammalian gene, and a simple, yet important eukaryote, yeast (*S. Cerevisiae*). The DNA HMAC structure preserves the intent of the design to mimic a genomic transcriptional control structure.

A successful, in vivo instantiation of a DNA HMAC system will require specific stop codons, start codons, promoters and enhancers sequences. An in vivo DNA encryption system should be multi-dimensional, utilize primary, secondary and tertiary structural information and include up/downstream regulators such that a single sequence can be seamlessly implemented at the genomic level and have multiple levels of encryption at the message or data level, depending upon the context (only known between sender and receiver). This approach also permits generation of mutant hash codes, which can be evaluated for fitness such that only the best hash code is selected for authentication purposes.

A. *Epigenetic relationships between cryptography and gene expression.*

Epigenetics involves heritable control of gene expression that does not involve modifications of the underlying DNA sequence [18]. Examples of epigenetic effects include: DNA methylation of cytosine residues, and control of gene expression via the higher order structures of DNA. In eukaryotes, DNA is packed into a hierarchy of structures: nucleosomes → chromatin → chromosomes Chromatin states can also be utilized as a form of encryption

and decryption by exposing or not exposing genes for transcription. Examples include:

- Heterochromatin form (encrypted) and Euchromatin form (decrypted) .
- Transcriptional memory via modification of chromatin states [19].
- Histone Code. A complex series of regulatory activities, which include histone lysine acetylation by histone acetyl transferase – transcriptionally active chromatin (decrypted); Histone lysine deacetylation by histone deacetylase – transcriptionally inactive (encrypted) [20], [21].

Expansion of the cryptographic protocols to include epigenetic operations will increase the richness of the protocols and the options for producing combinations of cipherproteins.

VIII. CONCLUSION

A cryptographic hash code based upon a DNA alphabet and a secure MANET authentication protocol has been presented. These codes can be utilized at the network level or application level and can also be implemented directly into genomes of choice to provide a new level of ciphertext communication at the genomic and proteomic level. The DNA inspired cryptographic coding approach is an option in developing true MANET architectures and developing novel forms of biological authentication to augment those architectures.

ACKNOWLEDGMENT

Thanks to the NASA Space Network, NASA/GSFC Exploration and Space Communications Projects Division, and the NASA Space Communications and Navigation Program office for supporting this research.

REFERENCES

[1] H. Shaw, S. Hussein, and H. Helgert, "Prototype Genomics-Based Keyed-Hash Message Authentication Code Protocol," 2nd International Conference on Evolving Internet, pp. 131-136, September 2010, doi: 10.1109/INTERNET.2010.31

[2] A. Gehani, T. LaBean, and J. Reif, "DNA-based Cryptography, Aspects of Molecular Computing", Springer-Verlag Lecture Notes in Computer Science, vol. 2950, pp. 167—188, 2004.

[3] N.G. Bourbakis, "Image Data Compression-Encryption Using G-Scan Patterns", Systems, Man, and Cybernetics, IEEE International Conference on Computational Cybernetics and Simulation, vol. 2, pp. 1117—1120, October 1997, doi: 10.1109/ICSMC.1997.638099

[4] A. Leier, C. Richter, W. Banzhaf, and H. Rauhe, "Cryptography with DNA binary strands", BioSystems, vol. 57, issue 1, pp. 13-22, June 2000, doi:10.1016/S0303-2647(00)00083-6

[5] C.T. Clelland, V. Risca, and C. Bancroft, "Hiding Messages in DNA microdots", Nature, vol. 399, pp. 533—534, June 1999, doi:10.1038/21092

[6] D. Heider and A. Barnekow, "DNA-based watermarks using the DNA-Crypt algorithm", BMC Bioinformatics, vol. 8, pp. 176, May 2007, doi:10.1186/1471-2105-8-176

[7] D. Heider and A. Barnekow, "DNA watermarks: A proof .of concept", BMC Molecular Biology 2008, vol. 9, p, 40, doi:10.1186/1471-2199-9-40

[8] H. Shaw and S. Hussein, "A DNA-Inspired Encryption Methodology for Secure, Mobile Ad-Hoc Networks (MANET)", Proceedings of the First International Conference on Biomedical Electronics and Devices, BIOSIGNALS 2008, Funchal, Madeira, Portugal, vol. 2, pp. 472-477, January 28-31, 2008

[9] Functional and Comparative Genomics Fact Sheet, Human GenomeProject, September 19, 2008, Available: http://www.ornl.gov/sci/techresources/Human_Genome/faq/compgen.shtml, accessed: May 31, 2011

[10] B. Alberts, A. Johnson, J. Lewis, L. Raff, K. Roberts, and P. Walter, Molecular Biology of the Cell, 4th edition , New York, NY: Garland Science, 2002

[11] W. Stallings, Cryptography and Network Security, 4th edition, Upper Saddle River, NJ: Pearson Prentice-Hall , 2006

[12] C. M. Fraser, J. D. Gocayne, O. White, M.D Adams, R. A. Clayton, R. D. Fleischmann, et al., "The Minimal Gene Complement of *Mycoplasma genitalium*", Science, vol. 270, No. 5235, pp. 397-403, Oct. 20, 1995, doi: 10.1126/science.270.5235.397

[13] C. J. Mitchell, "Truncation attacks on MACs", Electronics Letters - IET, vol. 39, part 20, pp 1439-1440, 2003, doi: 10.1049/el:20030921

[14] S. F. Elena, P. Carrasco, J. A. Daròs, and R. Sanjuán, "Mechanisms of genetic robustness in RNA viruses", EMBO Report, vol. 7, pp. 168-173, doi: 10.1038/sj.embor.7400636

[15] M. W. Nachman and S. L. Crowell., "Estimate of the mutation rate per nucleotide in humans.", Genetics, vol. 156, pp. 297-304, September 2000, Genetics Society of America

[16] C. R. Reeves and J. E. Rowe, Genetic Algorithms - Principles and Perspectives : A Guide to GA Theory, Dordrecht, Netherlands, Kluwer Academic Publishers, 2002

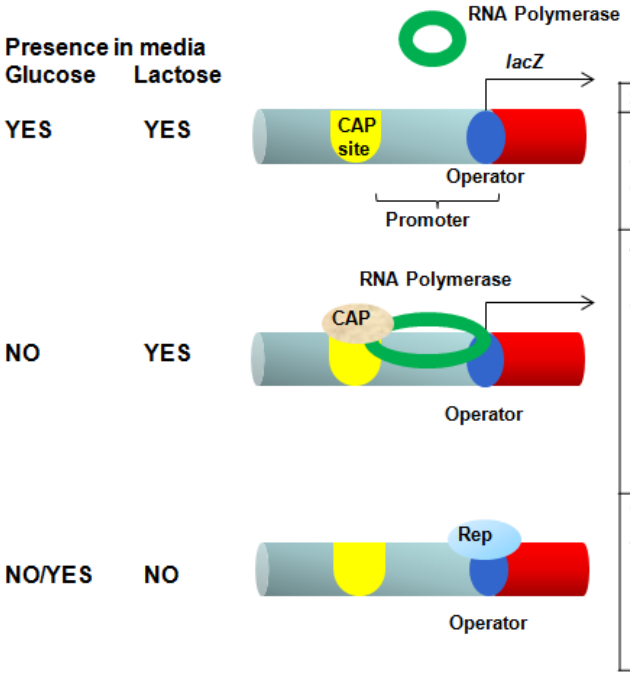
[17] C. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, p. 623, July 1948

[18] M. Ptashne and A. Gann , Genes and Signals, Cold Spring Harbor, NY, Cold Spring Harbor Laboratory Press, 2001

[19] S. Kundu and C. L. Peterson, "Role of chromatin states in transcriptional memory", Biochim Biophys Acta., vol. 1790, issue 6, pp: 445–455, June 2009, doi:10.1016/j.bbagen.2009.02.009

[20] S. Thiagalingam, K.-H. Cheng, H. J. Lee, N. Mineva, A. Thiagalingam, and J. F. Ponte, "Histone Deacetylases: Unique Players in Shaping the Epigenetic Histone Code", Annals of the New York Academy of Sciences, vol. 983: pp. 84–100, March 2003, doi: 10.1111/j.1749-6632.2003.tb05964.x

[21] T. Jenuwein and C. D. Allis, "Translating the Histone Code", Science, vol. 293, No. 5532, pp. 1074-1080, August 2001, doi: 10.1126/science.1063127



State	Operations	Comments
Basal (Encrypted) *lacZ* low level of expression	D(key $_{Lactose}$, *lacZ*) E(key $_{Glucose}$, *lacZ*) E(key $_{CAP}$, *lacZ*)	lacZ gene is encrypted by presence of glucose and lack of CAP at the *lacZ* promoter binding site
Activated (Decrypted) *lacZ* expressed	D(key $_{Lactose}$, *lacZ*) D(key $_{Glucose}$, *lacZ*) D(key $_{CAP}$, *lacZ*) MAC(key $_{CAPsite}$, *lacZ*) MAC($_{keyRNA\ Polyermase\ site}$, *lacZ*)	Absence of glucose and presence of CAP at *lacZ* promoter decrypts the *lacZ* gene. CAP and RNA polymerase MAC authenticate the binding sites so that RNA Polymerase can bind to *lacZ* promoter
Off (Encrypted) *lacZ* repressed by Rep protein	E(key $_{Lactose}$, *lacZ*) MAC (key $_{Rep\ site}$, *lacZ* E($_{keyRep}$, *lacZ*)	Absence of lactose allows Repressor protein to authenticate binding site to *lacZ* operator and encrypts *lacZ* gene by binding to operator site

Figure 12. Conceptual example of Confidentiality and Authentication in *E. coli* using *lacZ* expression

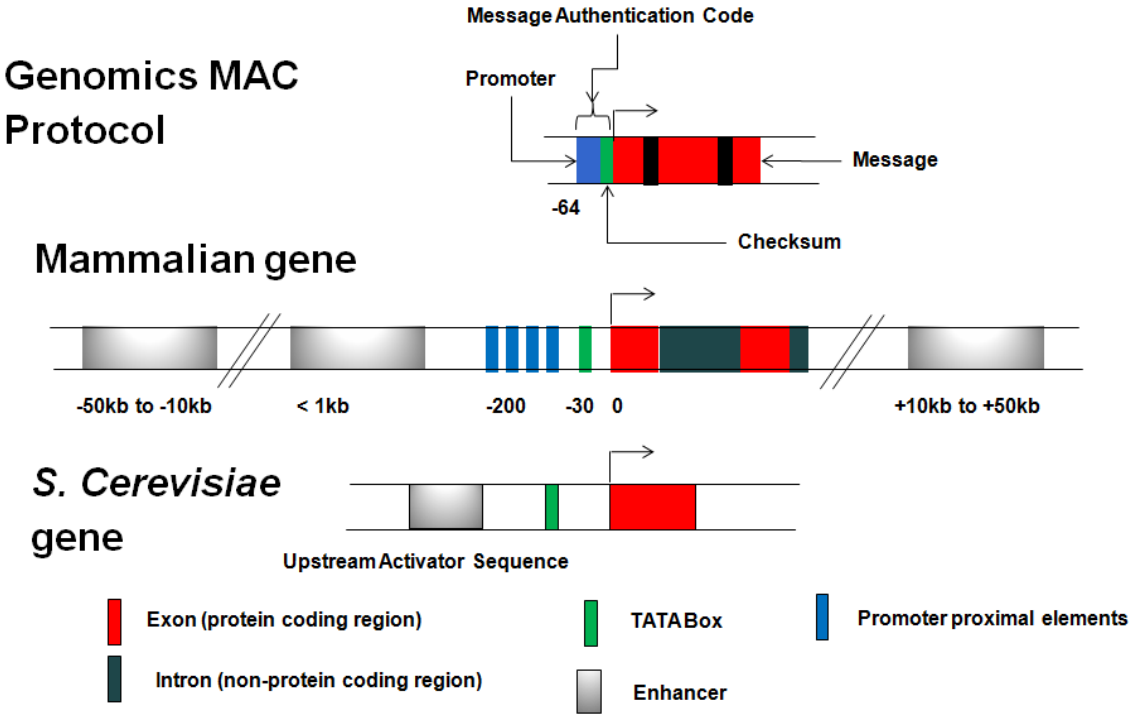


Figure 13. Simplified comparison between gene transcription control regions and MAC protocol

Evaluating Quality of Chaotic Pseudo-Random Generators: Application to Information Hiding

Jacques M. Bahi, Xiaole Fang, Christophe Guyeux, and Qianxue Wang
University of Franche-Comté
Computer Science Laboratory LIFC, Besançon, France
Email:{jacques.bahi, xiaole.fang, christophe.guyeux, qianxue.wang}@univ-fcomte.fr

***Abstract*—Guaranteeing the security of information transmitted through the Internet, against passive or active attacks, is a major concern. The discovery of new pseudo-random number generators with a strong level of security is a field of research in full expansion, due to the fact that numerous cryptosystems and data hiding schemes are directly dependent on the quality of these generators. At the conference Internet'09, we described a generator based on chaotic iterations which behaves chaotically as defined by Devaney. In this paper which is an extension of the work presented at the conference Internet'10, the proposal is to improve the speed, the security, and the evaluation of this generator, to make its use more relevant in the Internet security context. In order to do so, a comparative study between various generators is carried out and statistical results are improved. Finally, an application in the information hiding framework is presented with details, to give an illustrative example of the use of such a generator in the Internet security field.**

***Keywords*-Internet security; Pseudo-random number generator; Chaotic sequences; Statistical tests; Discrete chaotic iterations; Information hiding.**

I. Introduction

Due to the rapid development of the Internet in recent years, the need to find new tools to reinforce trust and security through the Internet has become a major concern. Its recent role in everyday life implies the need to protect data and privacy in digital world. This extremely rapid development of the Internet brings more and more attention to the information security techniques in all kinds of applications. For example, new security concerns have recently appeared because of the evolution of the Internet to support such activities as e-Voting, VoD (Video on demand), and the protection of intellectual property. In all these emerging techniques, pseudo-random number generators (PRNG) play an important role, because they are fundamental components of almost all cryptosystems and information hiding schemes [2], [3]. PRNGs are typically defined by a deterministic recurrent sequence in a finite state space, usually a finite field or ring, and an output function mapping each state to an input value. Following [4], this value is often either a real number in the interval $(0, 1)$ or an integer in some finite range. PRNGs based on linear congruential methods and feedback shift-registers are popular for historical reasons [5], but their security level often has been revealed to be inadequate by today's standards. However, to use a PRNG with a high level of security is a necessity to protect the information contents sent through the Internet. This level depends both on theoretical properties and on statistical tests.

Many PRNGs have already been proven to be secure following a probabilistic approach [6], [7], [8]. However, their performances must regularly be improved, among other things by using new mathematical tools. This is why the idea of using chaotic dynamical systems for this purpose has recently been explored [9], [10]. The random-like and unpredictable dynamics of chaotic systems, their inherent determinism and simplicity of realization suggest their potential for exploitation as PRNGs. Such generators can strongly improve the confidence put in any information hiding scheme and in cryptography in general: due to their properties of unpredictability, the possibilities offered to an attacker to achieve his goal are drastically reduced in that context. For example, in cryptography, keys are needed to be unpredictable enough, to make sure any search optimization based on the reduction of the key space to the most probable values is impossible to work on. But the number of generators claimed as chaotic, which actually have been proven to be unpredictable (as it is defined in the mathematical theory of chaos) is very small.

II. Outline of our Work

This paper extends the study initiated in [1], [11], [12], and tries to fill this gap. In [11], it is mathematically proven that chaotic iterations (CIs), a suitable tool for fast computing distributed algorithms, satisfies the topological chaotic property, following the definition given by Devaney [13]. In the paper [12] presented at Internet'09, the chaotic behavior of CIs is exploited in order to obtain an unpredictable PRNG that depends on two logistic maps. We have shown that, in addition to being chaotic, this generator can pass the NIST (National Institute of Standards and Technology of the U.S. Government) battery of tests [14], widely considered as a comprehensive and stringent battery of tests for cryptographic applications. In this paper, which is an extension of [1], we have improved the speed, security, and evaluation of the former generator and of its application in information hiding. Chaotic properties, statistical tests, and security analysis [15] allow us to consider that this generator has good characteristics and is capable to withstand attacks. After having presented the theoretical framework of the study and a security analysis, we will give a comparison based on statistical tests. Finally a concrete example of how to use these pseudo-random numbers for information hiding through the Internet is detailed.

The remainder of this paper is organized in the following way. In Section III, some basic definitions concerning chaotic iterations and PRNGs are recalled. Then, the generator based on discrete chaotic iterations is presented in Section IV. Section V is devoted to its security analysis. In Section VI, various tests are passed with a goal to achieve a statistical comparison between this new PRNG and other existing ones.

In Section VII, a potential use of this PRNG in some Internet security field is presented, namely in information hiding. The paper ends with a conclusion and intended future work.

III. Review of Basics

A. Notations

$[\![1;\mathsf{N}]\!]$ $\rightarrow \{1,2,\ldots,\mathsf{N}\}$
S^n $\rightarrow$ the n^{th} term of a sequence $S=(S^1,S^2,\ldots)$
v_i $\rightarrow$ the i^{th} component of a vector $v=(v_1,v_2,\ldots,v_n)$
f^k $\rightarrow$ k^{th} composition of a function f
strategy $\rightarrow$ a sequence which elements belong in $[\![1;\mathsf{N}]\!]$
$\mathbb{S}$ $\rightarrow$ the set of all strategies
$\mathbf{C}_n^k$ $\rightarrow$ the binomial coefficient $\binom{n}{k}=\frac{n!}{k!(n-k)!}$
$\oplus$ $\rightarrow$ bitwise exclusive or
$+$ $\rightarrow$ the integer addition
$\ll$ and $\gg$ $\rightarrow$ the usual shift operators
$(\mathcal{X},\mathrm{d})$ $\rightarrow$ a metric space
mod $\rightarrow$ a modulo or remainder operator
$\lfloor x \rfloor \rightarrow$ returns the highest integer smaller than x
$n! \rightarrow$ the factorial $n!=n\times(n-1)\times\cdots\times 1$
$\mathbb{N}^* \rightarrow$ the set of positive integers {1,2,3,...}

B. XORshift

XORshift is a category of very fast PRNGs designed by George Marsaglia [16]. It repeatedly uses the transform of exclusive or (XOR) on a number with a bit shifted version of it. The state of a XORshift generator is a vector of bits. At each step, the next state is obtained by applying a given number of XORshift operations to w-bit blocks in the current state, where $w=32$ or 64. A XORshift operation is defined as follows. Replace the w-bit block by a bitwise XOR of the original block, with a shifted copy of itself by a positions either to the right or to the left, where $0<a<w$. This Algorithm 1 has a period of $2^{32}-1=4.29\times10^9$.

Input: the internal state z (a 32-bit word)
Output: y (a 32-bit word)
$z \leftarrow z \oplus (z \ll 13)$;
$z \leftarrow z \oplus (z \gg 17)$;
$z \leftarrow z \oplus (z \ll 5)$;
$y \leftarrow z$;
return y;

Algorithm 1: An arbitrary round of XORshift algorithm

C. Continuous Chaos in Digital Computers

In the past two decades, the use of chaotic systems in the design of cryptosystems, pseudo-random number generators (PRNG), and hash functions, has become more and more frequent. Generally speaking, the chaos theory in the continuous field is used to analyze performances of related systems. However, when chaotic systems are realized in digital computers with finite computing precisions, it is doubtful whether or not they can still preserve the desired dynamics of the continuous chaotic systems. Because most dynamical properties of chaos are meaningful only when dynamical systems evolve in the continuous phase space, these properties may become meaningless or ambiguous when the phase space is highly quantized (i.e., latticed) with a finite computing precision (in other words, dynamical degradation of continuous chaotic systems realized in finite computing precision). When chaotic systems are realized in finite precision, their dynamical properties will be deeply different from the properties of continuous-value systems and some dynamical degradation will arise, such as short cycle length and decayed distribution. This phenomenon has been reported and analyzed in various situations [17], [18], [19], [20], [21].

Therefore, continuous chaos may collapse into the digital world and the ideal way to generate pseudo-random sequences is to use a discrete-time chaotic system.

D. Chaos for Discrete Dynamical Systems

Consider a metric space $(\mathcal{X},d)$ and a continuous function $f:\mathcal{X}\longrightarrow\mathcal{X}$, for one-dimensional dynamical systems of the form:

$$x^0\in\mathcal{X} \text{ and } \forall n\in\mathbb{N}^*, x^n=f(x^{n-1}), \tag{1}$$

the following definition of chaotic behavior, formulated by Devaney [13], is widely accepted:

Definition 1 A dynamical system of Form (1) is said to be chaotic if the following conditions hold.

- Topological transitivity:
$$\forall U,V \text{ open sets of } \mathcal{X}\setminus\varnothing,\ \exists k>0, f^k(U)\cap V\neq\varnothing \tag{2}$$
- Density of periodic points in $\mathcal{X}$:
Let $P=\{p\in\mathcal{X}|\exists n\in\mathbb{N}^*: f^n(p)=p\}$ the set of periodic points of f. Then P is dense in $\mathcal{X}$:
$$\overline{P}=\mathcal{X} \tag{3}$$
- Sensitive dependence on initial conditions: $\exists\varepsilon>0$, $\forall x\in\mathcal{X}$, $\forall\delta>0$, $\exists y\in\mathcal{X}$, $\exists n\in\mathbb{N}$, $d(x,y)<\delta$ and $d\left(f^n(x),f^n(y)\right)\geqslant\varepsilon$.

When f is chaotic, then the system $(\mathcal{X},f)$ is chaotic and quoting Devaney: "it is unpredictable because of the sensitive dependence on initial conditions. It cannot be broken down or decomposed into two subsystems which do not interact because of topological transitivity. And, in the midst of this random behavior, we nevertheless have an element of regularity." Fundamentally different behaviors are consequently possible and occur in an unpredictable way.

E. Discrete Chaotic Iterations

Definition 2 The set $\mathbb{B}$ denoting $\{0,1\}$, let $f:\mathbb{B}^{\mathsf{N}}\longrightarrow\mathbb{B}^{\mathsf{N}}$ be an "iteration" function and $S\in\mathbb{S}$ be a chaotic strategy. Then, the so-called *chaotic iterations* [22] are defined by $x^0\in\mathbb{B}^{\mathsf{N}}$ and

$$\forall n\in\mathbb{N}^*,\forall i\in[\![1;\mathsf{N}]\!], x_i^n=\begin{cases} x_i^{n-1} & \text{if } S^n\neq i \\ f(x^{n-1})_{S^n} & \text{if } S^n=i. \end{cases} \tag{4}$$

In other words, at the n^{th} iteration, only the S^n-th cell is "iterated". Note that in a more general formulation, S^n can be a subset of components and $f(x^{n-1})_{S^n}$ can be replaced by $f(x^k)_{S^n}$, where $k<n$, describing for example delays transmission. For the general definition of such chaotic iterations, see, e.g., [22].

Chaotic iterations generate a set of vectors (Boolean vector in this paper), they are defined by an initial state x^0, an iteration function f, and a chaotic strategy S.

The next section gives the outline proof that chaotic iterations satisfy Devaney's topological chaos property. Thus they can be used to define a chaotic pseudo-random bit generator.

IV. The Generation of CI Pseudo-Random Sequence

A. A Theoretical Proof for Devaney's Chaotic Dynamical Systems

The outline proofs, of the properties on which our pseudo-random number generator is based, are given in this section.

Denote by δ the *discrete Boolean metric*, $\delta(x,y) = 0 \Leftrightarrow x = y$. Given a function f, define the function $F_f : [\![1;\mathsf{N}]\!] \times \mathbb{B}^{\mathsf{N}} \longrightarrow \mathbb{B}^{\mathsf{N}}$ such that

$$F_f(k,E) = \left(E_j.\delta(k,j) + f(E)_k.\overline{\delta(k,j)}\right)_{j\in[\![1;\mathsf{N}]\!]},$$

where + and . are the Boolean addition and product operations.

Consider the phase space: $\mathcal{X} = [\![1;\mathsf{N}]\!]^{\mathbb{N}} \times \mathbb{B}^{\mathsf{N}}$ and the map

$$G_f\left(S,E\right) = \left(\sigma(S), F_f(i(S),E)\right),$$

then the chaotic iterations defined in (III-E) can be described by the following iterations [11]

$$\left\{\begin{array}{l} X^0 \in \mathcal{X} \\ X^{k+1} = G_f(X^k). \end{array}\right.$$

Let us define a new distance between two points $(S,E),(\check{S},\check{E}) \in \mathcal{X}$ by

$$d((S,E);(\check{S},\check{E})) = d_e(E,\check{E}) + d_s(S,\check{S}),$$

where

- $d_e(E,\check{E}) = \sum_{k=1}^{\mathsf{N}} \delta(E_k,\check{E}_k) \in [\![0;\mathsf{N}]\!]$
- $d_s(S,\check{S}) = \frac{9}{\mathsf{N}} \sum_{k=1}^{\infty} \frac{|S^k - \check{S}^k|}{10^k} \in [0;1].$

It is then proven in [11] by using the sequential continuity that

Proposition 1 *G_f is a continuous function on $(\mathcal{X},d)$.*

Then, the vectorial negation $f_0(x_1,\ldots,x_{\mathsf{N}}) = (\overline{x_1},\ldots,\overline{x_{\mathsf{N}}})$ satisfies the three conditions for Devaney's chaos, namely, regularity, transitivity, and sensitivity in the metric space $(\mathcal{X},d)$. This leads to the following result.

Proposition 2 *G_{f_0} is a chaotic map on $(\mathcal{X},d)$ in the sense of Devaney.*

B. Chaotic Iterations as Pseudo-Random Generator

1) Presentation: The CI generator (generator based on chaotic iterations) is designed by the following process. First of all, some chaotic iterations have to be done to generate a sequence $(x^n)_{n\in\mathbb{N}} \in \left(\mathbb{B}^{\mathsf{N}}\right)^{\mathbb{N}}$ ($\mathsf{N} \in \mathbb{N}^*, \mathsf{N} \geqslant 2$, N is not necessarily equal to 32) of Boolean vectors, which are the successive states of the iterated system. Some of these vectors will be randomly extracted and our pseudo-random bit flow will be constituted by their components. Such chaotic iterations are realized as follows. Initial state $x^0 \in \mathbb{B}^{\mathsf{N}}$ is a Boolean vector taken as a seed (see Section IV-B2) and chaotic strategy $(S^n)_{n\in\mathbb{N}} \in [\![1,\mathsf{N}]\!]^{\mathbb{N}}$ is an irregular decimation of a XORshift sequence (Section IV-B4). The iterate function f is the vectorial Boolean negation:

$$f_0 : (x_1,...,x_{\mathsf{N}}) \in \mathbb{B}^{\mathsf{N}} \longmapsto (\overline{x_1},...,\overline{x_{\mathsf{N}}}) \in \mathbb{B}^{\mathsf{N}}.$$

At each iteration, only the S^i-th component of state $\underline{x^n}$ is updated, as follows: $x_i^n = x_i^{n-1}$ if $i \neq S^i$, else $x_i^n = \overline{x_i^{n-1}}$. Finally, some x^n are selected by a sequence m^n as the pseudo-random bit sequence of our generator. $(m^n)_{n\in\mathbb{N}} \in \mathcal{M}^{\mathbb{N}}$ is computed from a XORshift sequence $(y^n)_{n\in\mathbb{N}} \in [\![0, 2^{32}-1]\!]$ (see Section IV-B3). So, the generator returns the following values:

Bits:

$$x_1^{m_0} x_2^{m_0} x_3^{m_0} \ldots x_{\mathsf{N}}^{m_0} x_1^{m_0+m_1} x_2^{m_0+m_1} \ldots x_{\mathsf{N}}^{m_0+m_1} x_1^{m_0+m_1+m_2} \ldots$$

or States:

$$x^{m_0} x^{m_0+m_1} x^{m_0+m_1+m_2} \ldots$$

2) The seed: The unpredictability of random sequences is established using a random seed that is obtained by a physical source like timings of keystrokes. Without the seed, the attacker must not be able to make any predictions about the output bits, even when all details of the generator are known [23].

The initial state of the system x^0 and the first term y^0 of the XORshift are seeded either by the current time in seconds since the Epoch, or by a number that the user inputs. Different ways are possible. For example, let us denote by t the decimal part of the current time. So x^0 can be t (mod 2^N) written in binary digits and $y^0 = t$.

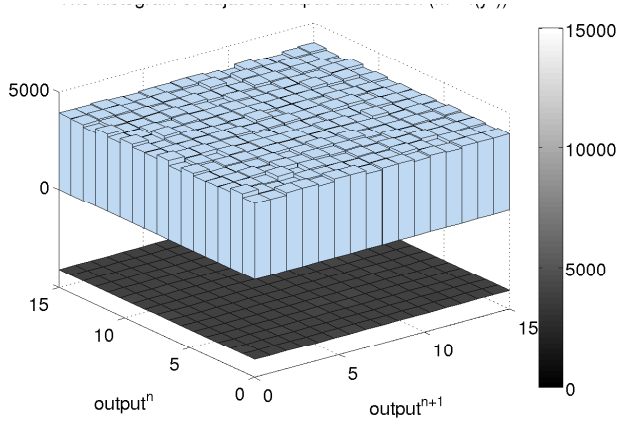
3) Sequence m of returned states: The output of the sequence (y^n) is uniform in $[\![0, 2^{32}-1]\!]$, because it is produced by a XORshift generator. However, we do not want the output of (m^n) to be uniform in $[\![0, N]\!]$, because in this case, the returns of our generator will not be uniform in $[\![0, 2^{\mathsf{N}}-1]\!]$, as it is illustrated in the following example. Let us suppose that $x^0 = (0,0,0)$. Then $m^0 \in [\![0,3]\!]$.

- If $m^0 = 0$, then no bit will change between the first and the second output of our PRNG. Thus $x^1 = (0,0,0)$.
- If $m^0 = 1$, then exactly one bit will change, which leads to three possible values for x^1, namely $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$.
- *etc.*

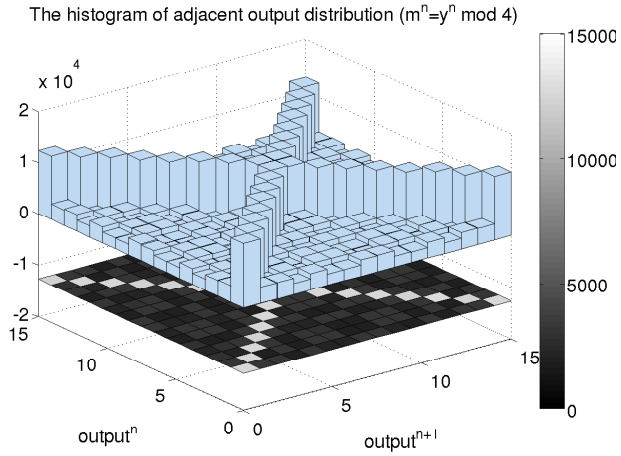
As each value in $[\![0, 2^3-1]\!]$ must be returned with the same probability, then the values $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$ must occur for x^1 with the same probability. Finally we see that, in this example, $m^0 = 1$ must be three times more probable than $m^0 = 0$. This leads to the following general definition for m:

$$m^n = g_1(y^n) = \left\{\begin{array}{l} 0 \text{ if } 0 \leqslant \frac{y^n}{2^{32}} < \frac{C_{\mathsf{N}}^0}{2^{\mathsf{N}}}, \\ 1 \text{ if } \frac{C_{\mathsf{N}}^0}{2^{\mathsf{N}}} \leqslant \frac{y^n}{2^{32}} < \sum_{i=0}^{1} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}}, \\ 2 \text{ if } \sum_{i=0}^{1} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}} \leqslant \frac{y^n}{2^{32}} < \sum_{i=0}^{2} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}}, \\ \vdots \quad\quad \vdots \\ \mathsf{N} \text{ if } \sum_{i=0}^{\mathsf{N}-1} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}} \leqslant \frac{y^n}{2^{32}} < 1. \end{array}\right. \quad (5)$$

In order to evaluate our proposed method and compare its statistical properties with various other methods, the density histogram and intensity map of adjacent outputs have been computed. The length of x is $\mathsf{N} = 4$ bits, and the initial conditions and control parameters are the same. A large number of sampled values are simulated (10^6 samples). Figure 1(a) shows



(a) $m^n = f(y^n)$



(b) $m^n = y^n \ mod \ 4$

Fig. 1: Histogram and intensity maps

the intensity map for $m^n = g_1(y^n)$. In order to appear random, the histogram should be uniformly distributed in all areas. It can be observed that a uniform histogram and a flat color intensity map are obtained when using our scheme. Another illustration of this fact is given by Figure 1(b), whereas its uniformity is further justified by the tests presented in Section VI.

4) Chaotic strategy: The chaotic strategy $(S^k) \in [\![1, \mathsf{N}]\!]^{\mathbb{N}}$ is generated from a second XORshift sequence $(b^k) \in [\![1, N]\!]^{\mathbb{N}}$. The only difference between the sequences S and b is that some terms of b are discarded, in such a way that $\forall k \in \mathbb{N}, (S^{M^k}, S^{M^k+1}, \ldots, S^{M^{k+1}-1})$ does not contain any given integer twice, where $M^k = \sum_{i=0}^{k} m^i$. Therefore, no bit will change more than once between two successive outputs of our PRNG, increasing the speed of the former generator by doing so. S is said to be "an irregular decimation" of b. This decimation can be obtained by the following process.

Let $(d^1, d^2, \ldots, d^{\mathsf{N}}) \in \{0,1\}^{\mathsf{N}}$ be a mark sequence, such that whenever $\sum_{i=1}^{\mathsf{N}} d^i = m^k$, then $\forall i, d_i = 0$ ($\forall k$, the sequence is reset when d contains m^k times the number 1). This mark sequence will control the XORshift sequence b as follows:

- if $d^{b^j} \neq 1$, then $S^k = b^j$, $d^{b^j} = 1$, and $k = k + 1$,
- if $d^{b^j} = 1$, then b^j is discarded.

For example, if $b = 1422334142112234...$ and $m = 4341...$, then $S = 1423\ 341\ 4123\ 4...$ However, if we do not use the mark sequence, then one position may change more than once and the balance property will not be checked, due to the fact that $\bar{\bar{x}} = x$. As an example, for b and m as in the previous example, $S = 1422\ 334\ 1421\ 1...$ and $S = 14\ 4\ 42\ 1...$ lead to the same outputs (because switching the same bit twice leads to the same state).

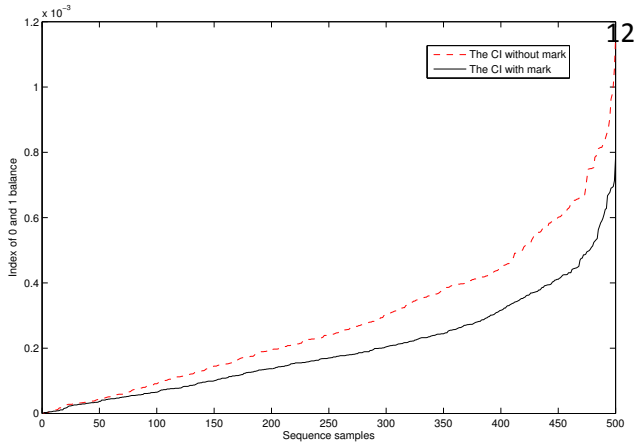


Fig. 2: Balance property

To check the balance property, a set of 500 sequences are generated with and without decimation, each sequence containing 10^6 bits. Figure 2 shows the percentages of differences between zeros and ones, and presents a better balance property for the sequences with decimation. This claim will be verified in the tests section (Section VI).

Another example is given in Table I, in which r means "reset" and the integers which are underlined in sequence b are discarded.

C. CI(XORshift, XORshift) Algorithm

The basic design procedure of the novel generator is summed up in Algorithm 2. The internal state is x, the output state is r. a and b are those computed by the two XORshift generators. The value $g_1(a)$ is an integer, defined as in Equation 5. Lastly, N is a constant defined by the user.

As a comparison, the basic design procedure of the old generator is recalled in Algorithm 3 (a and b are computed by logistic maps, N and $c \geqslant 3\mathsf{N}$ are constants defined by the user). See [12] for further information.

D. Illustrative Example

In this example, $\mathsf{N} = 4$ is chosen for easy understanding. As stated before, the initial state of the system x^0 can be seeded by the decimal part t of the current time. For example, if the current time in seconds since the Epoch is 1237632934.484088, so $t = 484088$, then $x^0 = t \ (mod\ 16)$ in binary digits, *i.e.*, $x^0 = (0, 1, 0, 0)$.

To compute m sequence, Equation 5 can be adapted to this example as follows:

$$m^n = g_1(y^n) = \begin{cases} 0 & \text{if} \quad 0 \leqslant \frac{y^n}{2^{32}} < \frac{1}{16}, \\ 1 & \text{if} \quad \frac{1}{16} \leqslant \frac{y^n}{2^{32}} < \frac{5}{16}, \\ 2 & \text{if} \quad \frac{5}{16} \leqslant \frac{y^n}{2^{32}} < \frac{11}{16}, \\ 3 & \text{if} \quad \frac{11}{16} \leqslant \frac{y^n}{2^{32}} < \frac{15}{16}, \\ 4 & \text{if} \quad \frac{15}{16} \leqslant \frac{y^n}{2^{32}} < 1, \end{cases} \tag{6}$$

Input: the internal state x (N bits)
Output: a state r of N bits
for $i = 0, \ldots,$ N **do**
  $d_i \leftarrow 0$;
end
$a \leftarrow XORshift1()$;
$m \leftarrow g_1(a)$;
$k \leftarrow m$;
for $i = 0, \ldots, k$ **do**
  $b \leftarrow XORshift2()\ mod$ N;
  $S \leftarrow b$;
  if $d_S = 0$ **then**
    $x_S \leftarrow \overline{x_S}$;
    $d_S \leftarrow 1$;
  end
  else if $d_S = 1$ **then**
    $k \leftarrow k + 1$;
  end
end
$r \leftarrow x$;
return r;

Algorithm 2: An arbitrary round of the new CI(XORshift,XORshift) generator

Input: the internal state x (N bits)
Output: a state r of N bits
$a \leftarrow Logisticmap1()$;
if $a > 0.5$ **then**
  $d \leftarrow 1$
end
else
  $d \leftarrow 0$
end
$m \leftarrow d + c$;
for $i = 0, \ldots, m$ **do**
  $b \leftarrow Logisticmap2()$;
  $S \leftarrow 100000b\ mod$ N;
  $x_S \leftarrow \overline{x_S}$;
end
$r \leftarrow x$;
return r;

Algorithm 3: An arbitrary round of the old CI PRNG

where y is generated by XORshift seeded with the current time. We can see that the probabilities of occurrences of $m = 0$, $m = 1$, $m = 2$, $m = 3$, $m = 4$, are $\frac{1}{16}$, $\frac{4}{16}$, $\frac{6}{16}$, $\frac{4}{16}$, $\frac{1}{16}$, respectively. This m determines what will be the next output x. For instance,

- If $m = 0$, the following x will be $(0, 1, 0, 0)$.
- If $m = 1$, the following x can be $(1, 1, 0, 0)$, $(0, 0, 0, 0)$, $(0, 1, 1, 0)$, or $(0, 1, 0, 1)$.
- If $m = 2$, the following x can be $(1, 0, 0, 0)$, $(1, 1, 1, 0)$, $(1, 1, 0, 1)$, $(0, 0, 1, 0)$, $(0, 0, 0, 1)$, or $(0, 1, 1, 1)$.
- If $m = 3$, the following x can be $(0, 0, 1, 1)$, $(1, 1, 1, 1)$, $(1, 0, 0, 1)$, or $(1, 0, 1, 0)$.
- If $m = 4$, the following x will be $(1, 0, 1, 1)$.

In this simulation, $m = 0, 4, 2, 2, 3, 4, 1, 1, 2, 3, 0, 1, 4, ...$ Additionally, b is computed with a XORshift generator too, but with another seed. We have found $b = 1, 4, 2, 2, 3, 3, 4, 1, 1, 4, 3, 2, 1, ...$

Chaotic iterations are made with initial state x^0, vectorial logical negation f_0, and strategy S. The result is presented in Table I. Let us recall that sequence m gives the states x^n to return, which are here $x^0, x^{0+4}, x^{0+4+2}, \ldots$ So, in this example, the output of the generator is: 10100111101111110011... or 4,4,11,8,1...

V. Security Analysis

PRNG should be sensitive with respect to the secret key and its size. Here, chaotic properties are also in close relation with the security.

A. Key Space

The PRNG proposed in this paper is based on discrete chaotic iterations. It has an initial value $x^0 \in \mathbb{B}^{\mathsf{N}}$. Considering this set of initial values alone, the key space size is equal to 2^{N}. In addition, this new generator combines digits of two other PRNGs. We used two different XORshifts here. Let k be the key space of XORshift, so the total key space size is close to $2^{\mathsf{N}} \cdot k^2$. Lastly, the impact of Equation 5, in which is defined the (m^n) sequence with a selector function g_1, must be taken into account. This leads to conclude that the key space size is large enough to withstand attacks.

Let us notice, to conclude this subsection, that our PRNG can use any reasonable function as selector. In this paper, $g_1()$ and $g_2()$ are adopted for demonstration purposes, where:

$$m^n = g_2(y^n) = \begin{cases} \mathsf{N} \text{ if } 0 \leqslant \frac{y^n}{2^{32}} < \frac{C_{\mathsf{N}}^0}{2^{\mathsf{N}}}, \\ \mathsf{N}-1 \text{ if } \frac{C_N^0}{2^{\mathsf{N}}} \leqslant \frac{y^n}{2^{32}} < \sum_{i=0}^{1} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}}, \\ \mathsf{N}-2 \text{ if } \sum_{i=0}^{1} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}} \leqslant \frac{y^n}{2^{32}} < \sum_{i=0}^{2} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}}, \\ \vdots \quad \vdots \\ 0 \text{ if } \sum_{i=0}^{\mathsf{N}-1} \frac{C_{\mathsf{N}}^i}{2^{\mathsf{N}}} \leqslant \frac{y^n}{2^{32}} < 1. \end{cases} \tag{7}$$

We will show later that both of them can pass all of the performed tests.

B. Key Sensitivity

As a consequence of its chaotic property, this PRNG is highly sensitive to the initial conditions. To illustrate this fact, several initial values are put into the chaotic system. Let H be the number of differences between the sequences obtained in this way. Suppose n is the length of these sequences. Then the variance ratio P, defined by $P = H/n$, is computed. The results are shown in Figure 3 (x axis is sequence lengths, y axis is variance ratio P). For the two PRNGs, variance ratios approach 0.50, which indicates that the system is extremely sensitive to the initial conditions.

C. Linear Complexity

The linear complexity (LC) of a sequence is the size in bits of the shortest linear feedback shift register (LFSR) which can produce this sequence. This value measures the difficulty of generating – and perhaps analyzing – a particular sequence. Indeed, the randomness of a given sequence can be linked to the size of the smallest program that can produce it. LC is the size required by a LFSR to be able to produce the given sequence. The Berlekamp-Massey algorithm can measure this

k	0	4			+1			2			2	+1		
b		1	4	2	<u>2</u>	3		3	4		1	<u>1</u>	4	
d	r	r $(1,0,0,0)^T$	$(1,0,0,1)^T$	$(1,1,0,1)^T$		$(1,1,1,1)^T$		r $(0,0,1,0)^T$	$(0,0,1,1)^T$		r $(1,0,0,0)^T$		$(1,0,0,1)^T$	
S		1	4	2		3		3	4		1		4	
x^0	x^0						x^4			x^6				x^8
0	0	$\xrightarrow{1} 1$					1			1	$\xrightarrow{1} 0$			0
1	1			$\xrightarrow{2} 0$			0			0				0
0	0					$\xrightarrow{3} 1$	1	$\xrightarrow{3} 0$		0				0
0	0		$\xrightarrow{4} 1$				1		$\xrightarrow{4} 0$	0			$\xrightarrow{4} 1$	1

Binary Output: $x_1^0x_2^0x_3^0x_4^0x_1^4x_2^4x_3^4x_4^4x_1^6x_2^6... = 0100101110000001...$
Integer Output: $x^0, x^4, x^6, x^8... = 4, 11, 8, 1...$

TABLE I: Example of New CI(XORshift,XORshift) generation

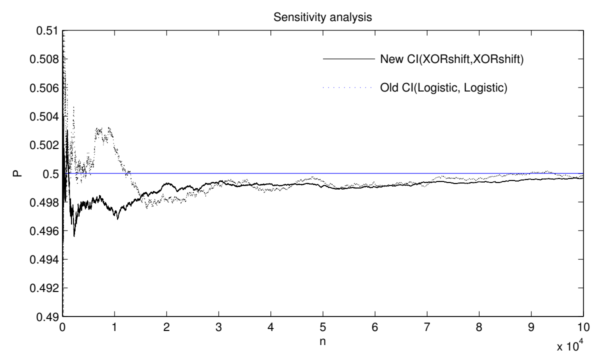


Fig. 3: Sensitivity analysis

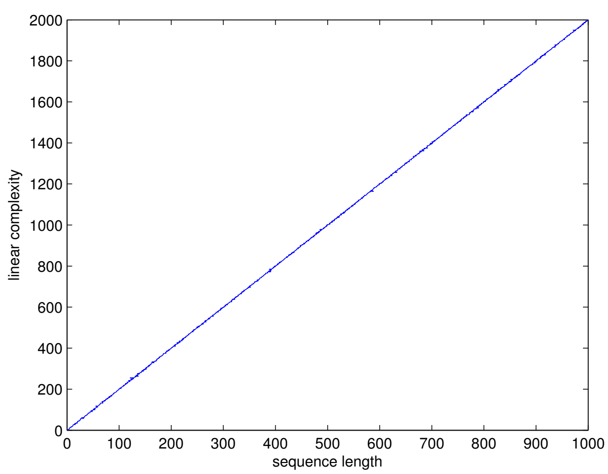


Fig. 4: Linear complexity

LC, which can be used to evaluate the security of a pseudo-random sequence. It can be seen in Figure 4 that the LC curve of a sample sequence of 2000 bits is close to the ideal line $C_i = i/2$, which implies that the generator has high linear complexity.

D. Devaney's Chaos Property

Generally speaking, the quality of a PRNG depends, to a large extent, on the following criteria: randomness, uniformity, independence, storage efficiency, and reproducibility. A chaotic sequence may satisfy these requirements and also other chaotic properties, as ergodicity, entropy, and expansivity. A chaotic sequence is extremely sensitive to the initial conditions. That is, even a minute difference in the initial state of the system can lead to enormous differences in the final state, even over fairly small timescales. Therefore, chaotic sequence fits the requirements of pseudo-random sequence well. Contrary to XORshift, our generator possesses these chaotic properties [11],[12]. However, despite a large number of papers published in the field of chaos-based pseudo-random generators, the impact of this research is rather marginal. This is due to the following reasons: almost all PRNG algorithms using chaos are based on dynamical systems defined on continuous sets (*e.g.*, the set of real numbers). So these generators are usually slow, requiring considerably more storage space, and lose their chaotic properties during computations as mentioned earlier in this paper. These major problems restrict their use as generators [24].

In this paper, we do not simply integrate chaotic maps hoping that the implemented algorithm remains chaotic. Indeed, the PRNG we conceive is just discrete chaotic iterations and we have proven in [11] that these iterations produce a topological chaos as defined by Devaney: they are regular, transitive, and sensitive to initial conditions. This famous definition of a chaotic behavior for a dynamical system implies unpredictability, mixture, sensitivity, and uniform repartition. Moreover, as only integers are manipulated in discrete chaotic iterations, the chaotic behavior of the system is preserved during computations, and these computations are fast.

Let us now explore the topological properties of our generator and their consequences concerning the quality of the generated pseudo-random sequences.

E. Topological Consequences

We have proven in [25] that chaotic iterations are expansive and topologically mixing. These topological properties are inherited by the generators we presented here. In particular, any error on the seed are magnified until being equal to the constant of expansivity. We will now investigate the consequences of being chaotic, as defined by Devaney.

First of all, the transitivity property implies the indecomposability of the system:

Definition 3 A dynamical system $(\mathcal{X}, f)$ is indecomposable if it is not the union of two closed sets $A, B \subset \mathcal{X}$ such that $f(A) \subset A, f(B) \subset B$.

Thus it is impossible to reduce the set of the outputs generated by our PRNG, in order to reduce its complexity. Moreover, it is possible to show that Old and New CI generators are strongly transitive:

Definition 4 A dynamical system $(\mathcal{X}, f)$ is strongly transitive if $\forall x, y \in \mathcal{X}$, $\forall r > 0$, $\exists z \in \mathcal{X}$, $d(z, x) \leqslant r \Rightarrow \exists n \in \mathbb{N}^*$, $f^n(z) = y$.

In other words, for all $x, y \in \mathcal{X}$, it is possible to find a point z in the neighborhood of x such that an iterate $f^n(z)$ is y. Indeed, this result has been established during the proof of the transitivity presented in [11]. Among other things, the strong transitivity property leads to the fact that without the knowledge of the seed, all of the outputs are possible. Additionally, no point of the output space can be discarded when studying our PRNG: it is intrinsically complicated and it cannot be simplified.

Finally, these generators possess the instability property:

Definition 5 A dynamical system $(\mathcal{X}, f)$ is unstable if for all $x \in \mathcal{X}$, the orbit $\gamma_x : n \in \mathbb{N} \longmapsto f^n(x)$ is unstable, that is: $\exists \varepsilon > 0$, $\forall \delta > 0$, $\exists y \in \mathcal{X}$, $\exists n \in \mathbb{N}$, $d(x, y) < \delta$ and $d\left(\gamma_x(n), \gamma_y(n)\right) \geqslant \varepsilon$.

This property, which is implied by the sensitive dependence to the initial condition, leads to the fact that in all of the neighborhoods of any x, there are points that are separate from x under iterations of f. We thus can claim that the behavior of our generators is unstable.

VI. Statistical Analysis

A. Basic Common Tests

1) Comparative test parameters: In this section, five well-known statistical tests [26] are used as comparison tools. They encompass frequency and autocorrelation tests. In what follows, $s = s^0, s^1, s^2, \ldots, s^{n-1}$ denotes a binary sequence of length n. The question is to determine whether this sequence possesses some specific characteristics that a truly random sequence would be likely to exhibit. The tests are introduced in this subsection and results are given in the next one.

a) Frequency test (monobit test): The purpose of this test is to check if the numbers of 0's and 1's are approximately equal in s, as it would be expected for a random sequence. Let n_0, n_1 denote these numbers. The statistic used here is:

$$X_1 = \frac{(n_0 - n_1)^2}{n},$$

which approximately follows a χ^2 distribution with one degree of freedom when $n \geqslant 10^7$.

b) Serial test (2-bit test): The purpose of this test is to determine if the number of occurrences of 00, 01, 10, and 11 as subsequences of s are approximately the same. Let n_{00}, n_{01}, n_{10}, and n_{11} denote the number of occurrences of $00, 01, 10$, and 11 respectively. Note that $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$ since the subsequences are allowed to overlap. The statistic used here is:

$$X_2 = \frac{4}{n-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{n}(n_0^2 + n_1^2) + 1,$$

which approximately follows a χ^2 distribution with 2 degrees of freedom if $n \geqslant 21$.

c) Poker test: The poker test studies if each pattern of length m (without overlapping) appears the same number of times in s. Let $\lfloor \frac{n}{m} \rfloor \geqslant 5 \times 2^m$ and $k = \lfloor \frac{n}{m} \rfloor$. Divide the sequence s into k non-overlapping parts, each of length m. Let n_i be the number of occurrences of the i^{th} type of sequence of length m, where $1 \leqslant i \leqslant 2^m$. The statistic used is

$$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k,$$

which approximately follows a χ^2 distribution with $2^m - 1$ degrees of freedom. Note that the poker test is a generalization of the frequency test: setting $m = 1$ in the poker test yields the frequency test.

d) Runs test: The purpose of the runs test is to figure out whether the number of runs of various lengths in the sequence s is as expected for a random sequence. A run is defined as a pattern of all zeros or all ones, a block is a run of ones, and a gap is a run of zeros. The expected number of gaps (or blocks) of length i in a random sequence of length n is $e_i = \frac{n-i+3}{2^{i+2}}$. Let k be equal to the largest integer i such that $e_i \geqslant 5$. Let B_i, G_i be the number of blocks and gaps of length i in s, for each $i \in [\![1, k]\!]$. The statistic used here will then be:

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i},$$

which approximately follows a χ^2 distribution with $2k - 2$ degrees of freedom.

e) Autocorrelation test: The purpose of this test is to check for coincidences between the sequence s and (non-cyclic) shifted versions of it. Let d be a fixed integer, $1 \leqslant d \leqslant \lfloor n/2 \rfloor$. The value $A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$ is the amount of bits not equal between the sequence and itself displaced by d bits. The statistic used here is:

$$X_5 = |2(A(d) - \frac{n-d}{2})/\sqrt{n-d}|,$$

which approximately follows a normal distribution $\mathcal{N}(0, 1)$ if $n - d \geqslant 10$. Since small values of $A(d)$ are as unexpected as large values, a two-sided test should be used.

2) Comparison: We show in Table II a comparison among our new generator CI(XORshift, XORshift), its old version denoted Old CI(Logistic, Logistic), a basic PRNG based on logistic map, and a simple XORshift. In this table, time (in seconds) is related to the duration needed by each algorithm to generate a 2×10^5 bits long sequence. The test has been conducted using the same computer and compiler with the same optimization settings for both algorithms, in order to make the test as fair as possible. The results confirm that the proposed generator is a lot faster than the old one, while the statistical results are better for most of the parameters, leading to the conclusion that the new PRNG is more secure than the old one. Although the logistic map also has good results, it is too slow to be implemented in Internet applications, and this map is known to present various bias leading to severe security issues.

As a comparison of the overall stability of these PRNGs, similar tests have been computed for different sequence lengths (see Figures 5 - 9). For the monobit test comparison (Figure 5),

TABLE II: Comparison with Old CI(Logistic, Logistic) for a 2×10^5 bits sequence

Method	Monobit (X_1)	Serial (X_2)	Poker (X_3)	Runs (X_4)	Autocorrelation (X_5)	Time
Logistic map	0.1280	0.1302	240.2893	26.5667	0.0373	0.965s
XORshift	1.7053	2.1466	248.9318	18.0087	0.5009	0.096s
Old CI(Logistic, Logistic)	1.0765	1.0796	258.1069	20.9272	1.6994	0.389s
New CI(XORshift,XORshift)	0.3328	0.7441	262.8173	16.7877	0.0805	0.197s

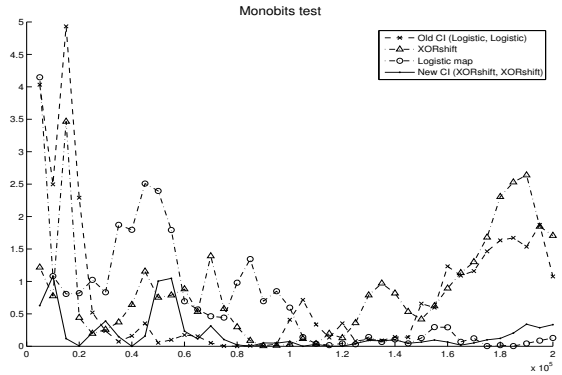


Fig. 5: Comparison of monobits tests

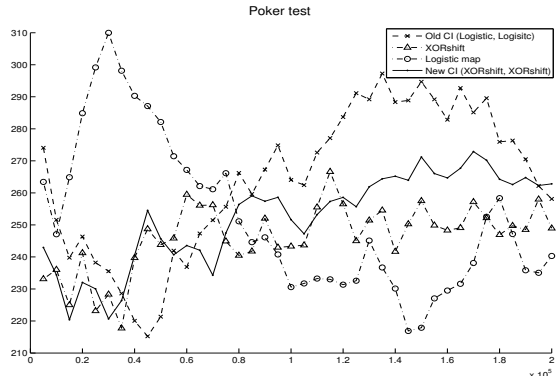


Fig. 7: Comparison of poker tests

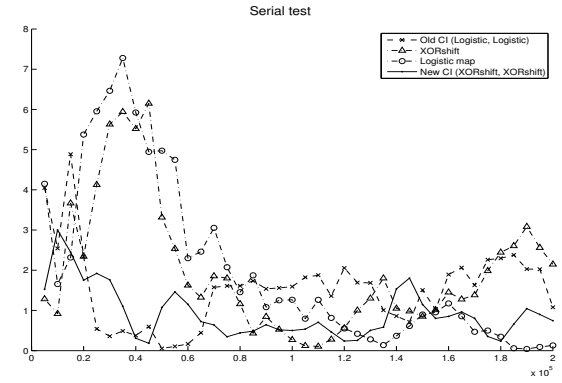


Fig. 6: Comparison of serial tests

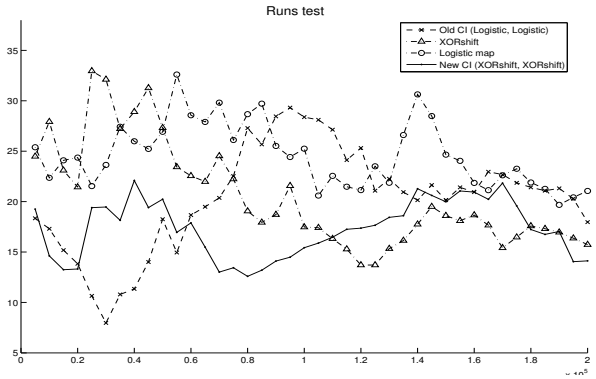


Fig. 8: Comparison of runs tests

almost all of the PRNGs present the same issue: the beginning values are a little high. However, for our new generator, the values are stable in a low level which never exceeds 1.2. Indeed, the new generator distributes very randomly the zeros and ones, whatever the length of the desired sequence. It can also be remarked that the old generator presents the second best performance, due to its use of chaotic iterations.

Figure 6 shows the serial test comparison. The new generator outperforms this test, but the score of the old generator is not bad either: their occurrences of 00, 01, 10, and 11 are very close to each other.

The poker test comparison with $m = 8$ is shown in Figure 7. XORshift is the most stable generator in all of these tests, and the logistic map also becomes good when producing sequences of length greater than 1×10^5. Our old and new generators present a similar trend, with a maximum in the neighborhood of 1.7×10^5. These scores are not so good, even though the new generator has a better behavior than the old one. Indeed, the value of m and the length of the sequences should be enlarged to be certain that the chaotic iterations express totally their complex behavior. In that situation, the performances of our generators in the poker test can be improved.

The graph of the new generator is the most stable one during the runs test comparison (Figure 8). Moreover, this trend is reinforced when the lengths of the tested sequences are increased.

The comparison of autocorrelation tests is presented in Figure 9. The new generator clearly dominates these tests, whereas the score of the old generator is surprisingly bad. This difference between two generators based on chaotic iterations

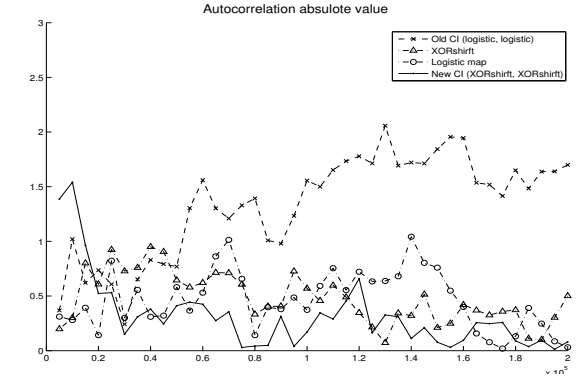


Fig. 9: Comparison of autocorrelation tests

can be explained by the fact that the improvements realized to define the new generator lead to a more randomly output.

To sum up we can claim that the new generator, which is faster than its former version, outperforms all of the other generators in these statistical tests, especially when producing long output sequences.

B. NIST Statistical Test Suite

1) Presentation: Among the numerous standard tests for pseudo-randomness, a convincing way to prove the quality of the produced sequences is to confront them with the NIST (National Institute of Standards and Technology) Statistical Test Suite SP 800-22, released by the Information Technology Laboratory in August 25, 2008.

The NIST test suite, SP 800-22, is a statistical package consisting of 15 tests. They were developed to measure the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic pseudo-random number generators. These tests focus on a variety of different types of non-randomness that could occur in such sequences. These 15 tests include in the NIST test suite are described in the Appendix.

2) Interpretation of empirical results: $\mathbb{P}$ is the "tail probability" that the chosen test statistic will assume values that are equal to or worse than the observed test statistic value when considering the null hypothesis. For each statistical test, a set of $\mathbb{P}$s is produced from a set of sequences obtained by our generator (i.e., 100 sequences are generated and tested, hence 100 $\mathbb{P}$s are produced).

Empirical results can be interpreted in various ways. In this paper, we check whether the $\mathbb{P}$s are uniformly distributed, via an application of a χ^2 distribution and the determination of a $\mathbb{P}_T$ corresponding to the Goodness-of-Fit distributional test on the $\mathbb{P}$s obtained for an arbitrary statistical test.

If $\mathbb{P}_T \geq 0.0001$, then the sequences can be considered to be uniformly distributed. In our experiments, 100 sequences (s = 100) of 1,000,000 bits are generated and tested. If the value $\mathbb{P}_T$ of a least one test is smaller than 0.0001, the sequences are considered to be not good enough and the generator is unsuitable.

Table III shows $\mathbb{P}_T$ for the sequences based on discrete chaotic iterations using different schemes. If there are at least two statistical values in a test, this test is marked with an asterisk and the average is computed to characterize the statistical values.

We can conclude from Table III that the worst situations are obtained with the New CI ($m^n = y^n \ mod \ N$) and New CI (no mark) generators. Old CI, New CI ($m^n = g_1(y^n)$), and New CI ($m^n = g_2(y^n)$) have successfully passed the NIST statistical test suite. These results and the conclusion obtained from the aforementioned basic tests reinforce the confidence that can be put in the good behavior of chaotic CI PRNGs, thus making them suitable for security applications as information hiding and digital watermarking.

VII. Application Example in Information Hiding

A. Introduction

Information hiding is now an integral part of Internet technologies. In the field of social search engines, for example, contents like pictures or movies are tagged with descriptive labels by contributors, and search results are determined by these descriptions. These collaborative taggings, used for example in Flickr [27] and Delicious [28] websites, contribute to the development of a Semantic Web, in which any Web page contains machine-readable metadata that describe its content. Information hiding technologies can be used for embedding these metadata. The advantage of its use is the possibility to realize social search without websites and databases: descriptions are directly embedded into media, whatever their formats. Robustness is required in this situation, as descriptions should resist to modifications like resizing, compression, and format conversion.

The Internet security field is also concerned by watermarking technologies. Steganography and cryptography are supposed to be used by terrorists to communicate through the Internet. Furthermore, in the areas of defense or in industrial espionage, many information leaks using steganographic techniques have been reported. Lastly, watermarking is often cited as a possible solution to digital rights managements issues, to counteract piracy of digital work in an Internet based entertainment world [29].

B. Definition of a Chaos-Based Information Hiding Scheme

Let us now introduce our information hiding scheme based on CI generator.

1) Most and least significant coefficients: Let us define the notions of most and least significant coefficients of an image.

Definition 1 For a given image, most significant coefficients (in short MSCs), are coefficients that allow the description of the relevant part of the image, *i.e.*, its richest part (in terms of embedding information), through a sequence of bits.

For example, in a spatial description of a grayscale image, a definition of MSCs can be the sequence constituted by the first four bits of each pixel (see Figure 10). In a discrete cosine frequency domain description, each 8×8 block of the carrier image is mapped onto a list of 64 coefficients. The energy of the image is mostly contained in a determined part of themselves, which can constitute a possible sequence of MSCs.

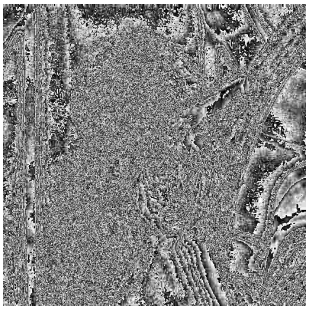
Definition 2 By least significant coefficients (LSCs), we mean a translation of some insignificant parts of a medium in a sequence of bits (insignificant can be understand as: "which can be altered without sensitive damages").

These LSCs can be, for example, the last three bits of the gray level of each pixel (see Figure 10). Discrete cosine, Fourier, and wavelet transforms can be used also to generate LSCs and MSCs. Moreover, these definitions can be extended to other types of media.

LSCs are used during the embedding stage. Indeed, some of the least significant coefficients of the carrier image will be chaotically chosen by using our PRNG. These bits will be either switched or replaced by the bits of the watermark. The MSCs are only useful in case of authentication; mixture and embedding stages depend on them. Hence, a coefficient should not be defined at the same time as a MSC and a LSC: the last can be altered while the first is needed to extract the watermark.

(a) Lena.

(b) MSCs of Lena.



(c) LSCs of Lena ($\times 17$).

Fig. 10: Example of most and least significant coefficients of Lena.

2) Stages of the scheme: Our CI generator-based information hiding scheme consists of two stages: (1) mixture of the watermark and (2) its embedding.

a) Watermark mixture: Firstly, for security reasons, the watermark can be mixed before its embedding into the image. A first way to achieve this stage is to apply the bitwise exclusive or (XOR) between the watermark and the New CI generator. In this paper, we introduce a new mixture scheme based on chaotic iterations. Its chaotic strategy, which depends on our PRNG, will be highly sensitive to the MSCs, in the case of an authenticated watermarking.

b) Watermark embedding: Some LSCs will be switched, or substituted by the bits of the possibly mixed watermark. To choose the sequence of LSCs to be altered, a number of integers, less than or equal to the number M of LSCs corresponding to a chaotic sequence U, is generated from the chaotic strategy used in the mixture stage. Thus, the U^k-th least significant coefficient of the carrier image is either switched, or substituted by the k^{th} bit of the possibly mixed watermark. In case of authentication, such a procedure leads to a choice of the LSCs that are highly dependent on the MSCs [30].

On the one hand, when the switch is chosen, the watermarked image is obtained from the original image whose LSBs $L = \mathbb{B}^{\mathsf{M}}$ are replaced by the result of some chaotic iterations. Here, the iterate function is the vectorial Boolean negation,

$$f_0 : (x_1, ..., x_{\mathsf{M}}) \in \mathbb{B}^{\mathsf{M}} \longmapsto (\overline{x_1}, ..., \overline{x_{\mathsf{M}}}) \in \mathbb{B}^{\mathsf{M}}, \qquad (8)$$

the initial state is L, and the strategy is equal to U. In this case, the whole embedding stage satisfies the topological chaos properties [30], but the original medium is required to extract the watermark. On the other hand, when the selected LSCs are substituted by the watermark, its extraction can be done without the original cover (blind watermarking). In this case, the selection of LSBs still remains chaotic because of the use of the New CI generator, but the whole process does not satisfy topological chaos [30]. The use of chaotic iterations is reduced to the mixture of the watermark. See the following sections for more detail.

c) Extraction: The chaotic strategy can be regenerated even in the case of an authenticated watermarking, because the MSCs have not changed during the embedding stage. Thus, the few altered LSCs can be found, the mixed watermark can be rebuilt, and the original watermark can be obtained. In case of a switch, the result of the previous chaotic iterations on the watermarked image should be the original cover. The probability of being watermarked decreases when the number of differences increase.

If the watermarked image is attacked, then the MSCs will change. Consequently, in case of authentication and due to the high sensitivity of our PRNG, the LSCs designed to receive the watermark will be completely different. Hence, the result of the recovery will have no similarity with the original watermark.

The chaos-based data hiding scheme is summed up in Figure 11.

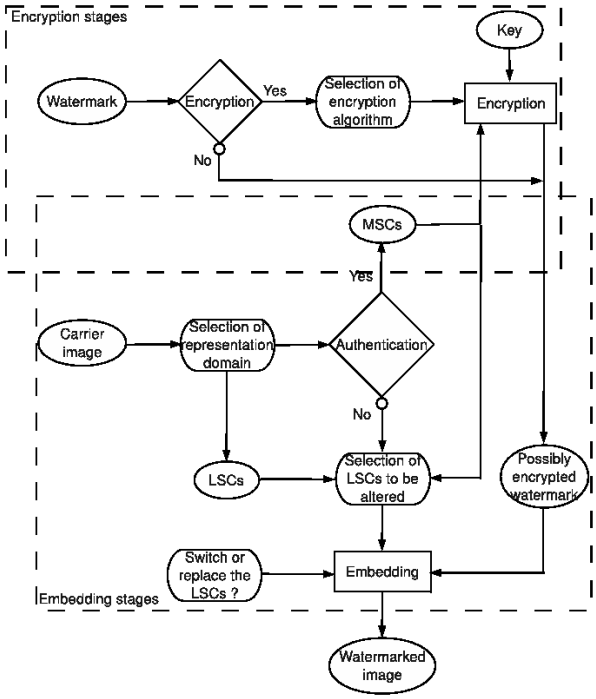


Fig. 11: The chaos-based data hiding decision tree.

C. Application Example

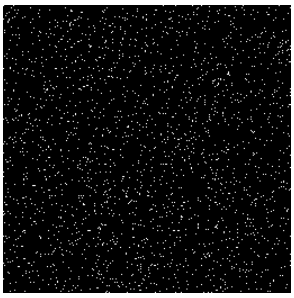
1) Experimental protocol: In this subsection, a concrete example is given: a watermark is encrypted and embedded into a cover image using the scheme presented in the previous section and CI(XORshift, XORshift). The carrier image is the well-known Lena, which is a 256 grayscale image, and the watermark is the 64×64 pixels binary image depicted in Figure 12.

The watermark is encrypted by using chaotic iterations: the initial state x^0 is the watermark, considered as a Boolean vector, the iteration function is the vectorial logical negation, and the chaotic strategy $(S^k)_{k\in\mathbb{N}}$ is defined with CI(XORshift, XORshift), where initial parameters constitute the secret key and $N = 64$. Thus, the encrypted watermark is the last

(a) The original image

(b) The watermark

Fig. 12: Original images



(a) Differences with the original

(b) Encrypted watermark

Fig. 13: Encrypted watermark and differences

Boolean vector generated by these chaotic iterations. An example of such an encryption is given in Figure 13.

Let L be the 256^3 Booleans vector constituted by the three last bits of each pixel of Lena and U^k defined by the sequence:

$$\begin{cases} U^0 &= S^0 \\ U^{n+1} &= S^{n+1} + 2 \times U^n + n \ [mod\ 256^3]. \end{cases} \quad (9)$$

The watermarked Lena I_w is obtained from the original Lena, whose three last bits are replaced by the result of 64^2 chaotic iterations with initial state L and strategy U (see Figure 13).

The extraction of the watermark can be obtained in the same way. Remark that the map $\theta \mapsto 2\theta$ of the torus, which is the famous dyadic transformation (a well-known example of topological chaos [13]), has been chosen to make $(U^k)_{k\leqslant 64^2}$ highly sensitive to the strategy. As a consequence, $(U^k)_{k\leqslant 64^2}$ is highly sensitive to the alteration of the image: any significant modification of the watermarked image will lead to a completely different extracted watermark, thus giving a way to authenticate media through the Internet.

Let us now evaluate the robustness of the proposed method.

2) Robustness evaluation: In what follows, the embedding domain is the spatial domain, CI(XORshift,XORshift) has been used to encrypt the watermark, MSCs are the four first bits of each pixel (useful only in case of authentication), and LSCs are the three next bits.

To prove the efficiency and the robustness of the proposed algorithm, some attacks are applied to our chaotic watermarked image. For each attack, a similarity percentage with the watermark is computed, this percentage is the number of equal bits between the original and the extracted watermark, shown as a percentage. Let us notice that a result less than or equal to 50% implies that the image has probably not been watermarked.

a) Zeroing attack: In this kind of attack, a watermarked image is zeroed, such as in Figure 14(a). In this case, the results in Table 1 have been obtained.

(a) Cropping attack

(b) Rotation attack

Fig. 14: Watermarked Lena after attacks.

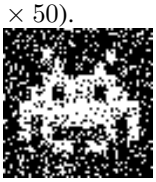
UNAUTHENTICATION		AUTHENTICATION	
Size (pixels)	Similarity	Size (pixels)	Similarity
10	99.08%	10	91.77%
50	97.31%	50	55.43%
100	92.43%	100	51.52%
200	70.75%	200	50.60%

Table. 1. Cropping attacks

In Figure 15, the decrypted watermarks are shown after a crop of 50 pixels and after a crop of 10 pixels, in the authentication case.

(a) Unauthentication (50×50).

(b) Authentication (50×50).



(c) Authentication (10×10).

Fig. 15: Extracted watermark after a cropping attack.

By analyzing the similarity percentage between the original and the extracted watermark, we can conclude that in case of unauthentication, the watermark still remains after a zeroing attack: the desired robustness is reached. It can be noticed that zeroing sizes and percentages are rather proportional.

In case of authentication, even a small change of the carrier image (a crop by 10×10 pixels) leads to a really different extracted watermark. In this case, any attempt to alter the carrier image will be signaled, the image is well authenticated.

b) Rotation attack: Let r_θ be the rotation of angle θ around the center $(128, 128)$ of the carrier image. So, the transformation $r_{-\theta} \circ r_\theta$ is applied to the watermarked image, which is altered as in Figure 14. The results in Table 2 have been obtained.

UNAUTHENTICATION		AUTHENTICATION	
Angle (degree)	Similarity	Angle (degree)	Similarity
2	96.44%	2	73.40%
5	93.32%	5	60.56%
10	90.68%	10	52.11%
25	78.13%	25	51.97%

Table. 2. Rotation attacks

TABLE III. SP 800-22 test results (1 T)

Method	New CI ($m^n = y^n \ mod \ N$)	New CI (no mark)	Old CI	New CI ($g_1()$)	New CI ($g_2()$)
Frequency (Monobit) Test	0.0004	0.0855	0.595549	0.474986	0.419
Frequency Test within a Block	0	0	0.554420	0.897763	0.6786
Runs Test	0.2896	0.5544	0.455937	0.816537	0.3345
Longest Run of Ones in a Block Test	0.0109	0.4372	0.016717	0.798139	0.8831
Binary Matrix Rank Test	0	0.6579	0.616305	0.262249	0.7597
Discrete Fourier Transform (Spectral) Test	0	0	0.000190	0.007160	0.0008
Non-overlapping Template Matching Test*	0.020071	0.37333	0.532252	0.449916	0.51879
Overlapping Template Matching Test	0	0	0.334538	0.514124	0.2492
Maurer's "Universal Statistical" Test	0.6993	0.9642	0.032923	0.678686	0.1296
Linear Complexity Test	0.3669	0.924	0.401199	0.657933	0.3504
Serial Test* (m=10)	0	0.28185	0.013396	0.425346	0.2549
Approximate Entropy Test (m=10)	0	0.3838	0.137282	0.637119	0.7597
Cumulative Sums (Cusum) Test*	0	0	0.046464	0.279680	0.34245
Random Excursions Test*	0.46769	0.34788	0.503622	0.287409	0.18977
Random Excursions Variant Test*	0.28779	0.46505	0.347772	0.486686	0.26563
Success	8/15	11/15	15/15	15/15	15/15

The same conclusion as above can be declaimed: this watermarking method satisfies the desired properties.

c) JPEG compression: A JPEG compression is applied to the watermarked image, depending on a compression level. Let us notice that this attack leads to a change of the representation domain (from spatial to DCT domain). In this case, the results in Table 3 have been obtained.

UNAUTHENTICATION		AUTHENTICATION	
Compression	Similarity	Compression	Similarity
2	85.76%	2	56.42%
5	67.62%	5	52.12%
10	62.43%	10	48.22%
20	54.74%	20	49.07%

Table. 3. JPEG compression attacks

A very good authentication through JPEG attack is obtained. As for the unauthentication case, the watermark still remains after a compression level equal to 10. This is a good result if we take into account the fact that we use spatial embedding.

d) Gaussian noise: Watermarked image can be also attacked by the addition of a Gaussian noise, depending on a standard deviation. In this case, the results in Table 4 have been obtained.

UNAUTHENTICATION		AUTHENTICATION	
Standard dev.	Similarity	Standard dev.	Similarity
1	81.14%	1	55.57%
2	75.01%	2	52.63%
3	67.64%	3	52.68%
5	57.48%	5	51.34%

Table. 4. Gaussian noise attacks

Once again we remark that good results are obtained, especially if we keep in mind that a spatial representation domain has been chosen.

VIII. Conclusion and Future Work

In this paper, the pseudo-random generator proposed in [12] has been improved. By using XORshift instead of logistic map and due to a rewrite of the way to generate strategies, the generator based on chaotic iterations works faster and is more secure. The speed and randomness of this new PRNG has been compared to its former version, to XORshift, and to a generator based on logistic map. This comparison shows that CI(XORshift, XORshift) offers a sufficient speed and level of security for a wide range of Internet usages as cryptography and information hiding.

In future work, we will continue to try to improve the speed and security of this PRNG, by exploring new strategies and iteration functions. Its chaotic behavior will be deepened by using the numerous tools provided by the mathematical theory of chaos. New statistical tests will be used to compare this PRNG to existing ones. Additionally a probabilistic study of its security will be done. Lastly, new applications in computer science will be proposed, especially in the Internet security field.

References

[1] Q. Wang, J. M. Bahi, C. Guyeux, and X. Fang, "Randomness quality of CI chaotic generators. application to internet security," in *INTERNET'2010. The 2nd Int. Conf. on Evolving Internet*. Valencia, Spain: IEEE seccion ESPANIA, Sep. 2010, pp. 125–130.

[2] X. Tong and M. Cui, "Image encryption scheme based on 3d baker with dynamical compound chaotic sequence cipher generator," *Signal Processing*, vol. 89, no. 4, pp. 480 – 491, 2009.

[3] E. Erclebi and A. SubasI, "Robust multi bit and high quality audio watermarking using pseudo-random sequences," *Computers Electrical Engineering*, vol. 31, no. 8, pp. 525 – 536, 2005.

[4] P. L'ecuyer, "Comparison of point sets and sequences for quasi-monte carlo and for random number generation," *SETA 2008*, vol. LNCS 5203, pp. 1–17, 2008.

[5] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Reading, Mass, and third edition, Eds. Addison-Wesley, 1998.

[6] A. Marchi, A. Liverani, and A. D. Giudice, "Polynomial pseudo-random number generator via cyclic phase," *Mathematics and Computers in Simulation*, vol. 79, no. 11, pp. 3328–3338, 2009.

[7] S. Sachez, R. Criado, and C. Vega, "A generator of pseudo-random numbers sequences with a very long period," *Mathematical and Computer Modelling*, vol. 42, pp. 809 – 816, 2005.

[8] C. J. K. Tan, "The plfg parallel pseudo-random number generator," *Future Generation Computer Systems*, vol. 18, no. 5, pp. 693 – 698, 2002.

[9] M. Falcioni, L. Palatella, S. Pigolotti, and A. Vulpiani, "Properties making a chaotic system a good pseudo random number generator," *arXiv*, vol. nlin/0503035, 2005.

[10] S. Cecen, R. M. Demirer, and C. Bayrak, "A new hybrid nonlinear congruential number generator based on higher functional power of logistic maps," *Chaos, Solitons and Fractals*, vol. 42, pp. 847–853, 2009.

[11] J. M. Bahi and C. Guyeux, "Topological chaos and chaotic iterations, application to hash functions," *WCCI'10: 2010 IEEE World Congress on Computational Intelligence*, vol. Accepted paper, 2010.
[12] Q. Wang, C. Guyeux, and J. M. Bahi, "A novel pseudo-random generator based on discrete chaotic iterations for cryptographic applications," *INTERNET '09*, pp. 71–76, 2009.
[13] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, 2nd ed. Redwood City: Addison-Wesley, 1989.
[14] N. S. Publication, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Aug. 2008.
[15] F. Zheng, X. Tian, J. Song, and X. Li, "Pseudo-random sequence generator based on the generalized henon map," *The Journal of China Universities of Posts and Telecommunications*, vol. 15(3), pp. 64–68, 2008.
[16] G. Marsaglia, "Xorshift rngs," *Journal of Statistical Software*, vol. 8(14), pp. 1–6, 2003.
[17] P. M. Binder and R. V. Jensen, "Simulating chaotic behavior with finite-state machines," *Physical Review A*, vol. 34, no. 5, pp. 4460–4463, 1986.
[18] D. D. Wheeler, "Problems with chaotic cryptosystems," *Cryptologia*, vol. XIII, no. 3, pp. 243–250, 1989.
[19] J. Palmore and C. Herring, "Computer arithmetic, chaos and fractals," *Physica D*, vol. 42, pp. 99–110, 1990.
[20] M. Blank, "Discreteness and continuity in problems of chaotic dynamics," *Translations of Mathematical Monographs*, vol. 161, 1997.
[21] S. Li, G. Chen, and X. Mou, "On the dynamical degradation of digital piecewise linear chaotic maps," *Bifurcation an Chaos*, vol. 15, no. 10, pp. 3119–3151, 2005.
[22] F. Robert, *Discrete Iterations. A Metric Study*. Springer Series in Computational Mathematics, 1986, vol. 6.
[23] M. S. Turan, A. Doganaksoy, and S. Boztas, "On independence and sensitivity of statistical randomness tests," *SETA 2008*, vol. LNCS 5203, pp. 18–29, 2008.
[24] L. Kocarev, "Chaos-based cryptography: a brief overview," *IEEE Circ Syst Mag*, vol. 7, pp. 6–21, 2001.
[25] C. Guyeux, N. Friot, and J. M. Bahi, "Chaotic iterations versus spread-spectrum: chaos and stego security," in *IIH-MSP'10, 6-th Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, Darmstadt, Germany, Oct. 2010, pp. 208–211, to appear.
[26] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of applied cryptography*, Bocarton, Ed. CRC Press, 1997.
[27] "The frick collection, http://www.frick.org/," Last visit the 7th of June, 2011.
[28] "Delicious social bookmarking, http://delicious.com/," Last visit the 7th of June, 2011.
[29] Y. Nakashima, R. Tachibana, and N. Babaguchi, "Watermarked movie soundtrack finds the position of the camcorder in a theater," *IEEE Transactions on Multimedia*, 2009, accepted for future publication Multimedia.
[30] J. M. Bahi and C. Guyeux, "Topological chaos and chaotic iterations, application to hash functions," in *WCCI'10, IEEE World Congress on Computational Intelligence*. Barcelona, Spain: IEEE, Jul. 2010, pp. 1–7.

APPENDIX

The NIST Statistical Test Suite

In what follows, the objectives of the fifteen tests contained in the NIST Statistical tests suite are recalled. A more detailed description for those tests can be found in [14].

Frequency (Monobit) Test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence.

Frequency Test within a Block is to determine whether the frequency of ones in an M-bits block is approximately M/2, as would be expected under an assumption of randomness (M is the length of each block).

Runs Test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow.

Test for the Longest Run of Ones in a Block is to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence.

Binary Matrix Rank Test is to check for linear dependence among fixed length substrings of the original sequence.

Discrete Fourier Transform (Spectral) Test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness.

Non-overlapping Template Matching Test is to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern.

Overlapping Template Matching Test is the number of occurrences of pre-specified target strings.

Maurer's "Universal Statistical" Test is to detect whether or not the sequence can be significantly compressed without loss of information.

Linear Complexity Test is to determine whether or not the sequence is complex enough to be considered random.

Serial Test is to determine whether the number of occurrences of the 2^m m-bit (m is the length in bits of each block) overlapping patterns is approximately the same as would be expected for a random sequence.

Approximate Entropy Test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence (m is the length of each block).

Cumulative Sums (Cusum) Test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences.

Random Excursions Test is to determine if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence.

Random Excursions Variant Test is to detect deviations from the expected number of visits to various states in the random walk.

Touch'n Trust: An NFC-Enabled Trusted Platform Module

Michael Hutter and Ronald Toegl
Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
Email:{michael.hutter,ronald.toegl}@iaik.tugraz.at

***Abstract*—Instant and ubiquitous access to devices such as public terminals raises several security concerns in terms of confidentiality and trust. While Trusted Computing introduces advanced security mechanisms into terminal hardware, there is often no convenient way to help users decide on the trustworthiness of a device. To overcome this issue, Near Field Communication (NFC) can be used to leverage the trusted-computing protocol of remote attestation. Here, NFC helps user to intuitively establish a communication between local devices. In this article, we propose an NFC-enabled Trusted Platform Module (TPM) architecture that allows users to verify the security status of public terminals. For this, we introduce an autonomic and low-cost NFC-compatible interface to the TPM to create a direct trusted channel. Users can access the TPM with NFC-enabled devices, which have become widely available in the form of smart phones. Elliptic-curve cryptography provides efficient signing and verifying of the security-status report. Furthermore, we implemented an NFC-enabled TPM platform as a proof-of-concept demonstrator and show that a trust decision can be realized with commodity mobile phones. It shows that an NFC-enabled TPM can effectively help to overcome confidentiality issues in common public-terminal applications.**

***Keywords*-Trusted Computing; RFID Security; Near Field Communication; NFC; ECDSA; Remote Attestation.**

I. Introduction

During the last decade, mobile technology has grown significantly offering many applications including payment, ticketing, and access control. Especially Near-Field Communication (NFC) has become widely available on the market offering convenient services by simply touching NFC-enabled objects in the proximity with commodity mobile phones. It is obvious that such ubiquitous communication raises concerns on the security and on the privacy, especially as NFC services are able to pinpoint the location of their users. Therefore, for users it would be desirable to only use NFC services that are worth of being entrusted with sensitive or personal information. In this article, which extends [1], we present a method to overcome confidentiality and trust issues in security-related NFC applications by taking advantage of Trusted Computing mechanisms.

Trusted Computing has been designed to provide services to establish trust in Internet environments. One such service, as devised by the Trusted Computing Group (TCG) industry consortium, is *remote attestation*. It achieves trust decisions between hosts. A TPM is therefore used to measure the integrity and confidentiality guaranteeing status of a targeted host platform. This state information must then be reported to the user who decides if the platform can be trusted or not. Unfortunately, the cryptographic protocols that actually perform the attestation are not suited for local and interactive service scenarios. They neither provide a human intelligible conveyance of the status report nor an intuitive identification of the targeted host platform. In a mobile user scenario, small and portable devices such as mobile phones or PDAs can help perform the attestation protocol reporting the status of the targeted platform to the user [11], [29], [47], [48]. It has also been established that to provide such a service, a trusted channel between the user and the platform-integrated TPM is mandatory [33].

In this article, we propose to integrate both an NFC interface and a TPM into a single hardware component. This integration provides a secure trust decision to the user by guaranteeing the physical presence of the user through NFC and offering the direct channel to the TPM by the combination of both modules in one piece of silicon. The proposed architecture leverages the remote attestation protocol so that a mobile phone can be used to establish trust to a public terminal in the proximity. We improve on a previously reported proposal [48] by moving the trust decision into the mobile device. This is enabled by substantial optimizations that include an advanced terminal hardware and software platform and a more efficient cryptographic protocol. To this end, we propose to use elliptic curve cryptography (ECC) to increase the computation and communication performance. A virtualization-based software architecture allows for compact and expressive state descriptions. As a proof-of-concept, we implemented an NFC-enabled TPM prototype that performs remote attestation using the elliptic curve digital signature algorithm (ECDSA). We further give performance results of our practical experiments, both in software and hardware.

The remainder of this article is structured as follows. In Section II, we give a short introduction into the NFC technology. Section III introduces Trusted Computing and the Trusted Platform Module. Section IV describes the proposed architecture that integrates an NFC-interface inside the TPM module. We present implementation details and performance results in Section V. The paper concludes in Section VI.

II. NFC Environments

NFC is a wireless communication technology that provides a platform for many applications such as mobile payment, ticketing, and access control. One key feature of NFC is the simple acquisition of data just by touching an object with an NFC-enabled reader. Such readers might be integrated in mobile phones or digital cameras that transfer information to the devices in their proximity. There exist two different modes for a communication between a reader (initiator) and a target device. In passive communication mode, the initiator provides an electromagnetic (EM) field which is used to power the target device and which allows a bidirectional communication. In active communication mode, both the initiator and the target device provide an alternately generated EM field so that both devices require an active power supply.

NFC is based on the Radio Frequency Identification (RFID) technology that operates at a frequency of 13.56 MHz. As opposed to RFID, NFC follows several specifications that have been standardized by the International Organization for Standardization (ISO) and the European Computer Manufacturers Association (ECMA). These standards specify the modulation, coding, frame formats, data rates, and also the application-specific transport protocols used. The typical operating distance between two NFC devices is only a few centimeters (up to 10 cm). Thus, installing an NFC device (passively or actively powered) to a fixed location can provide evidence whether a mobile NFC device (or its user) has been at that location or not. Besides this evidence, NFC offers a very intuitive way for the user to communicate with a target object by simply bringing the devices close together (touching). It therefore mimics the natural principle of communication between two locally present entities.

There already exist many mobile devices that support NFC. Typical examples are the 6131, 6212, and 3220 mobile phones from Nokia, the *SGH-X700* or *D500E* from Samsung, the *my700X* from SAGEM, the *L7* from Motorola, or the *T80* from Benq. NFC will be also an integral part of the next generation of smart-phones. Nokia's *C7* smart-phone already includes an NFC chip that can be activated by a software update in the first quarter of 2011. Another example is the *Nexus S* smart-phone from Google which has been available since December 2010. There have already been many companies that use that technology to provide different *touch and go* applications, *e.g.*, the *Touch and Travel* project of the German railway system and the mobile operator Vodafone, the e-ticket and e-payment services of Japan's mobile operator KDDI or NTT DoCoMo, or the Bay Area Rapid Transit (BART) system in the San Francisco Bay area.

The use and integration of NFC into next generation devices has been largely pushed and promoted by the NFC Forum since 2004. The NFC Forum is an association of about 140 industry partners such as NXP Semiconductors, Sony, Nokia, Microsoft, MasterCard, NEC, Visa, and Samsung. They encourage the development of new standards and products to ensure the interoperability of NFC-enabled devices. They defined four different types of NFC-Forum compatible devices. All types are based on the contact-less smart-card standard ISO 14443 [19] or FeliCa. Type 1 tags provide only simple functionalities such as writing of data into the memory while more complex type 4 tags typically involve also security features and support ISO/IEC 7816-4 Application Protocol Data Units (APDU).

However, as soon as entities use NFC services and establish a connection to terminals with their mobile phones, the questions of integrity and confidentiality rise inevitably. Especially in security-related applications like mobile payment and ticketing, cryptographic services are needed that provide a decision mechanism to the user if the connected device can be trusted or not. Compromised devices pose a serious thread that might extract secret information, perform unwanted payment transactions or behave untrustworthy in some other way.

III. Trusted Computing enabled Terminals

A. Trusted Platform Module

Trusted computing offers services to make trust decisions by integrating a so-called Trusted Platform Module (TPM) [45] into target machines, *e.g.*, public NFC terminals. The Trusted Computing Group (TCG) has specified the TPM for general purpose computer systems. Similar to a smart card, the TPM features cryptographic primitives but it is physically bound to its host device. A tamper-resilient integrated circuit contains implementations for public-key cryptography, key generation, cryptographic hashing, and random-number generation and provides therefore a root of trust.

In particular, the TPM implements high-level functionalities such as reporting the current system configuration and providing evidence of the integrity and authenticity of this measurement. This service is also known as *Remote Attestation*. During the remote attestation process, the TPM receives hashes of several system-state descriptors and stores the hashes in dedicated Platform Configuration Registers (PCRs) located in the TPM. In fact, a PCR with index i in state t is extended with input x by setting

$$PCR_i^{t+1} = \text{SHA1}(PCR_i^t || x).$$

The basic operation of a TPM is as follows. Before executable code is invoked, a hash value of the code is computed and stored in a PCR. Ultimately, if all components from the Basic Input/Output System (BIOS) up to a specific application are measured, the exact configuration of the platform is mapped to PCR values. This property makes it

impossible to hide a malicious program on a thus protected computer. If such a system state fulfills the given security or policy requirements, we refer to the system state as a *trusted state*.

The TPM can also *bind* data to the platform by encrypting it with a *non-migratable* key, which never leaves the TPM's protection. An extension to this is *sealing*, where a key may only be used with a specific PCR configuration. Thus, decryption of sealed data can be restricted to a trusted state of the computer. TPMs also provide a limited amount of non-volatile memory (NV-RAM) to store user- or owner-supplied information.

The TPM is capable of signing the current values of the PCRs together with a supplied nonce. This is called the *Quote* operation, which is the core operation in the *Remote Attestation* protocol [8], [40], [44]. To protect the platform owner's privacy, a pseudonym identity is used: an *Attestation Identity Key (AIK)*. The authenticity of an AIK can be certified either by an on-line trusted third party, called PrivacyCA [34] or by applying the group-signature-based DAA scheme [7], for instance. Then, a remote verifier may analyze the Quote result and decide whether to trust the given configuration or not.

The hardware resources of a TPM are manufacturer implementation specific and typically very limited. For instance, the TPM supplies only a few cryptographic key slots and continually swaps keys to and from external storage during operation. The current TPM design establishes the need for a singleton system software component to manage the TPM device resources and arbitrate concurrent accesses. To this end, the TCG specifies an architecture that implements TPM access and management, the *TCG Software Stack* (TSS) [46] which covers operating system and applications support.

B. Platform Virtualization for Attestation

Virtualization is a methodology of dividing the resources of a computer into multiple execution environments, by applying concepts such as time-sharing [43], hardware and software partitioning, machine simulation or emulation. Hardware architectures can be designed to offer complete virtualization [38] in hardware and then host unmodified operating systems in parallel. Only recently, the PC platform has been modified accordingly (see [2] for an overview).

Commonly, virtualization is controlled by a singleton *hypervisor*, a superior control entity which directly runs on the hardware and manages it exclusively. It enables the creation, execution and hibernation of isolated *partitions*, each hosting a guest operating system and the virtual applications building on it.

Such a system provides multiple isolation layers: Standard processor privilege rings and memory paging protect processes executing within a virtualization. Hardware support for monitoring all privileged CPU instructions enables the hypervisor to transparently isolate virtualization instances from each other. Finally, the chip-set is able to block direct memory accesses (DMA) of devices to defined physical memory areas, thus allowing the hypervisor to control device I/O.

Modern platforms from AMD [3] and Intel [12] extend the basic TCG model of a *static* chain-of-trust anchored in a hardware reboot. They provide the option of a *dynamic* switch to a trusted system state. In this paper we focus on Intel's *Trusted Execution Technology* (TXT), which we build our implementation on. Similar functionality is provided by AMD's Secure Virtual Machine (SVM).

A so-called *late launch* is initiated by the special Intel TXT CPU instruction `GETSEC[SENTER]`. It stops all processing cores except one. The chip-set locks all memory to prevent outside modification by DMA devices. A special Intel-provided and cryptographically signed *Authenticated Code Module* (ACM) starts a fresh chain-of-trust after setting the platform into a well-defined state. This provides a *Dynamic Root of Trust for Measurement* (DRTM).

Subsequently, a *Measured Launch Environment* (MLE) [18] is first measured and then executed. Piece-by-piece the MLE decides which system resources to unlock and thus cautiously restores normal platform operation. The platform remembers that she is running in "secrets" mode and automatically enforces memory scrubbing whenever a system (re)boot is initiated.

The ACM is also capable of enforcing specific *Launch Control Policies* (LCPs). Here, the ACM measures the MLE and compares it with the trusted LCP stored in the non-volatile memory of the TPM. Changes to the LCP can only be authorized by the TPM owner. Any other, not authorized software configuration is not allowed to continue; the ACM will reset the platform. This mechanism thus allows to perform a secure boot into a trusted state. Virtualization also allows to measure complete file system images that contain virtual applications. This leads to deterministic, yet expressive PCR values.

C. Remote Attestation over NFC

Attestation is useful to improve the security for a number of computing services, including not only remote but also physically present systems. In general, various types of systems may be encountered in different usage scenarios. For instance, a user might want to learn if a public general-purpose desktop computer is secure for ad-hoc use. Customers would like to be assured that a point-of-sales terminal in a shop will not collect their PIN together with the information on the magnetic stripe of their credit card for later frauds. The same holds true for other types of Automatic Teller Machines (ATMs) and payment terminals. Vending machines could be reconfigured by attackers to collect cash but not to release their goods. Other security critical applications may also be found in embedded systems or even peripherals like printers or access points. Here,

a service technician might find a method to identify the exact software configuration and its integrity to be useful. In addition, giving voters a method to validate that electronic voting machines have not been tampered might assist to add trust to a poll's outcome.

Public terminals are typically located at shops, in hotel lobbies, transportation terminals, or Internet cafés. They provide different services to users like Internet access via Web browsers or ticket-vending services. These terminals are publicly available and can be accessed by users several times always pretending a legitimate user. Possible attackers are therefore assumed to have full access over the software running on the terminals. As a result, the terminal can not be trusted. The users get no guarantee about the confidentiality and privacy status of the terminal they would like to use.

With TPM-based attestation, trust can be established between users and terminals. There exist several proposals for that. McCune et al. [29] pointed out that it would be desirable to equip the user with a simple, ideal and axiomatically trustworthy device. It would then indicate the security of a device to the user. Molnar et al. [30] describe an RFID-based solution. They proposed to integrate the remote attestation protocol into RFID readers to allow the verification of the privacy status of the reader to existing tags (or users) in the field. Li et al. [25] proposed to secure mobile-payment applications with remote attestation. They present practical results of an attestation scenario using NFC mobile phones. A similar approach has been presented by Garriss et al. [11]. They designed a protocol by which a mobile phone can determine the integrity of running software on a terminal (kiosk computer).

However, many of the proposed architectures suffer in the fact that the TCG's attestation protocol does not guarantee that the TPM is still located within the machine the user faces. This allows possible machine-in-the-middle attacks where an attacker establishes an indirect link between the user and the TPM over a distrusted channel.

IV. The NFC-Enabled TPM Architecture

In the following section, we briefly outline our design for an NFC-enabled TPM architecture. In contrast to existing publications, we propose to integrate the functionality of a low-cost passive RFID interface into the TPM. This allows users to remotely audit the privacy status of the terminal (target) using a conventional NFC device (initiator) by ensuring a direct channel between the user and the TPM. Figure 1 shows a schematic view of our proposed architecture. It shows the TPM device with its components such as a CPU, I/O interface, voltage regulator, memory, and cryptographic components like SHA-1, RSA/ECC, key generation, and a random number generation (RNG). Additionally, an RFID front-end is connected to the internal bus. It is composed of a digital part which handles the NFC protocol (ISO 14443 or ISO 18092) and an analog part that is mainly responsible to modulate and demodulate the signals of the air interface. Next to the I/O and power interface, the TPM has two additional connections for an external antenna that can be connected or printed on the main board of the terminal.

In general, passive RFID/NFC devices are designed to meet low-resource requirements. They draw their power of the air from the reader and therefore consume only a few microwatts of power to provide a certain reading range. Furthermore, they meet low area requirements so that they can be produced in a large scale with low costs. In fact, most RFID tags can nowadays be produced with costs below 10 cents.

TPMs, in contrast, have an active power supply and provide many resources such as cryptographic engines, CPU, and (in comparison) large memory units. These resources can actually be reused be the RFID/NFC front-end which lowers the requirements of integrating NFC into TPM significantly. The controlling of the protocol, for example, can be handled by the CPU of the TPM and memory can be shared with the RFID/NFC component over the internal bus architecture as well.

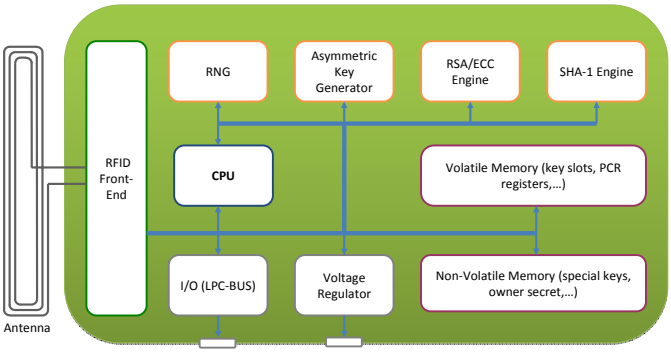


Figure 1. Schematic of the NFC-Enabled TPM architecture.

By integrating an NFC interface to the TPM, we followed the idea of Parno [33] who recommended to integrate a special-purpose hardware interface to the TPM to establish a direct link between the user and the TPM so that human inter-actors can themselves establish the proximity of the attesting machine. The integration of the NFC interface to the TPM has thereby several advantages.

First, a guarantee is given that the targeted terminal is still in the proximity of the user, as the user performs the touching operation in person.

Second, it gives a guarantee that the TPM is located in the proximity because the NFC module is physically connected and integrated into the TPM and the operational range of NFC is theoretical limited by the coupling property of the magnetic near-field of the reader. Note that RFID standards define a practical reading range of about 10 centimeters. This makes attacks such as machine-in-the-middle attacks much harder to perform.

Third, our proposal of integrating an NFC interface into the TPM allows users to verify the integrity of public terminals with their mobile phone. By simply touching the antenna of the terminal with their phone, an application can be automatically launched that shows the status of the trust decision on a user-friendly and familiar display output. In addition, the trust decision can be also signalized by a beep, ringing tone, or vibration which makes the application more applicable and comfortable in certain environments.

In the following, we describe the implemented trusted computing protocol of remote attestation that can be used to provide such a service.

A. Public Key Infrastructure

To support our software architecture, a Public Key Infrastructure (PKI) is needed. A PKI represents a system for binding a public key to a specific and distinct user or device identity by means of digital certificates. Certificates are signed by a Certification Authority after a Registration Authority established the identity and Revokation mechanisms blacklist compromised keys.

In our scheme a PrivacyCA acts as trusted third party for all participants. The AIK certificates it issues are created during the registration process of the unique terminal hardware platform. It ensures that a real hardware TPM is present on the registered hardware, and that the private part of each AIK will never be exposed by the TPM. Currently only Infineon TPMs can be verified because only they provide certificates for their Endorsement Keys.

Therefore, for every new terminal that is added to the network it is necessary that its TPM has been activated and the ownership has been taken. In the following we describe the registration process. At first, the identity of the hardware platform is established by qualified personnel. After the creation of the TPM-based AIK public/private key pair, the public part is sent to the PrivacyCA server together with the Endorsement Key certificate. The PrivacyCA server validates and analyses the EK certificate and decides whether to trust the TPM or not. If the server believes the TPM to be authentic, the server then certifies the AIK and encrypts the fresh certificate with the public Endorsement Key of the TPM. This is returned to the platform and can only be decrypted there. The AIK certificate is permanently stored at the PrivacyCA to allow a later revocation of this certificate.

B. The Remote Attestation Protocol

The first step in the attestation protocol is to generate a nonce N_0 with the NFC-enabled mobile phone which acts as a reader device (initiator). As soon as the mobile phone touches the RF antenna of the terminal, the nonce is transmitted over the air interface within a configuration challenge (the nonce serves as fresh data to avoid replay attacks). After that, the public terminal (target) responds with the Quote of its currently recorded terminal state. The Quote, *i.e.*, $Sig_{AIK}(PCR_n..PCR_m, N_0)$, $1 \leq n \leq m \leq 24$, the signature over the selected PCR registers under an Attestation Identity Key AIK of the TPM, is then returned to the mobile phone. The protocol flow is shown in Figure 2. Note that the figure shows only a very compact protocol flow that neither includes issues of key management nor the handling of certificates and trusted third party (TTP) services.

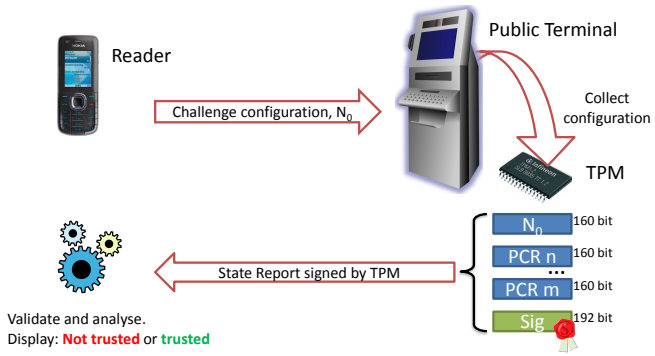


Figure 2. The compact NFC attestation protocol.

In order to sign the PCRs of the TPM prototype, we propose the use of elliptic curve cryptography (ECC). In contrast to other public-key primitives like RSA, ECC has gathered much attention due to the use of shorter key sizes. The computation time and especially the communication time over the air interface can be significantly improved by providing the same cryptographic strength. For instance, the strength of a 2 048 bit RSA key can be compared with that of a 224 bit ECC key.

Due to these reasons, we propose to use ECDSA to sign the data. The use of ECDSA has several advantages. First, the protocol has been standardized by several organizations such as ANSI [4], IEEE [16], NIST [32], and ISO/IEC [20]. Second, there already exist public-key infrastructures that support this algorithm for signing and verifying data and X.509 certificates [21].

The algorithm for generating digital signatures is shown in Algorithm 1. It takes a message m as input (containing N_0 in the TPM-Quote data structure) and outputs the digital signature (r, s) of that message. The private key d is securely stored in non-volatile memory. The most time consuming operation in ECDSA is the elliptic curve (EC) point multiplication $[k]P$ (line 2 in Algorithm 1). It takes more that 80 % of the total execution time. For this, a randomly generated value k (ephemeral key) is multiplied with a point P on an elliptic curve. The x-coordinate of the resulting point is then used further in the signing process. Next to that operation, a message digest algorithm (SHA-1) is used to hash the input message (line 4). The final signing process (line 5) needs several finite-field operations such as modular addition, modular multiplication, and modular inversion.

Algorithm 1 Signature generation using ECDSA

Require: Domain parameters, private key d, message m.
Ensure: Signature (r, s)

1: Select $k \in [1, n-1]$
2: Compute $[k]P = (x_1, y_1)$, convert x_1 to an integer $\bar{x}_1$
3: Compute $r = \bar{x}_1 \pmod n$. If $r = 0$ then go back to 1.
4: Compute $e = SHA1(m)$.
5: Compute $s = k^{-1}(e + dr) \pmod n$. If $s = 0$ then go back to step 1.
6: Return (r, s)

An ECDSA digital signature can be verified by the verification algorithm shown in Figure 2. First, the input signature (r, s) is verified to be in $[0, n-1]$ (line 1). After that, the hash of the message m is calculated (line 2). Line 3-4 show the calculation of the intermediate variables w, u_1, and u_2. In line 5, two point multiplications have to be performed resulting in the elliptic-curve point X. The signature is valid if the relation $v = r$ holds, otherwise it is rejected.

Algorithm 2 Signature verification using ECDSA

Require: Domain parameters, public key Q, message m, signature (r, s).
Ensure: Accept or Reject of (r, s)

1: Verify that $r, s \in [0, n-1]$.
2: Compute $e = SHA1(m)$.
3: Compute $w = s^{-1} \pmod n$.
4: Compute $u_1 = ew \pmod n$ and $u_2 = rw \pmod n$.
5: Compute $X = (x_1, y_1) = u_1 P + u_2 Q$.
6: If $X = \infty$ then return.
7: Compute $v = \bar{x}_1 \pmod n$.
8: If $v = r$ then $accept$ else $reject$.

If the signature has been validated successfully, the Quote information is compared to a list of known-good PCR values. A quote is only accepted if the state report contains only trusted values.

V. Implementation Results

In the following, we present results of an implementation of the proposed system architecture. First, we describe the implementation of a public terminal that runs on an attestation-friendly software platform. Second, we describe an NFC-enabled TPM prototype that can be touched by NFC-enabled mobile phones. Third, we show an attestation scenario example and give results about the implementation's performance.

A. The Public Terminal

As a public terminal, we used an attestation-friendly hardware platform that is based on the Intel Trusted Execution Technology (TXT). We measure and enforce the terminal integrity with the acTvSM software platform [35], [36], [49]. It relies on a TPM to provide basic Trusted Computing services such as secure storage of software measurements and on hardware-based virtualization to execute programs in a trusted environment. The terminal application is, together with its operating system, contained in an image file, which is measured before execution. From within the virtual machine, we can access the TPM using IAIK jTSS [37] to retrieve the Quote that reflects the system state.

1) Software Components: Secure boot is accomplished by using a standard boot-loader GRUB [10] along with `SINIT` and `tboot` [17]. `SINIT` is Intel's implementation of an ACM, while `tboot` is Intel's prototype implementation of an MLE. Upon boot GRUB loads SINIT, tboot, the kernel and its `initramfs` into memory and executes tboot, which sets up the ACM and then late-launches into it. The authenticity and integrity of the ACM code is guaranteed under an Intel private key, of which the public part is hardwired into the chip-set. The ACM's task is then to measure the tboot binary and compare it to the LCP. Tboot takes over and continues to measure the kernel and `initramfs` and compares them against the VLP. Once the integrity of the kernel and its environment has been assured, control is passed to it and the standard boot process continues. Customized 64-bit ports of tools from IBM's *TPM-utils* [15] provide the PCR extend and unsealing capabilities in the initial ram-disk (`initramfs`) environment.

In our architecture, we use a customized Linux operating system augmented with the *Kernel-based Virtual Machine* (KVM) [23], [24] hypervisor module. KVM can run multiple virtual machines on x86 CPUs equipped with virtualization mode extensions. It extends the Linux Kernel to offer, besides the existing Kernel and User modes, an additional Guest process mode. Each virtual machine offers private virtualized hardware like a network card, hard disk, graphics adapter, etc. Those virtual devices are forwarded to *QEMU* [6], a fast software emulator. QEMU can emulate all standard hardware devices of a x86 platform, including one or several processors. For the Base system, we use packages from the x86_64 *Debian Linux* lenny release [41]. It acts as the host for the virtualization partitions. To support current Trusted Computing and virtualization hardware we need to add selected packages from the Debian testing tree. For example, only Linux kernels 2.6.32 or newer integrate Intel TXT support and drivers for chip-sets that implement it. Scripts for installation, initial ram-disk management and rebuilding of the Base System image are taken from Debian and customized to our needs. The system bootstrap scripts for creation of distributable and boot-able CDs for initial installation are taken from GRML Linux [39], a distribution specialized for system administrators.

2) Application Image Management: We use a complex disk layout with different file systems to create a measurable platform.

A first partition contains a read-write file-system hosting all the components necessary for the platform boot process. This encompasses the boot-loader, `tboot`, `SINIT` and Linux kernel plus associated initramfs images. The remainder of the harddisk storage is allocated as a Logical Volume Manager (LVM) [27] dynamically managed space, which is assigned to a single LVM volume group.

The LVM managed volume group contains both plain-text and encrypted logical volumes (LVs). These block devices are transparently encrypted with Linux kernel's `dm-crypt` subsystem. Each LV contains either a file system for use on the platform, or raw storage which is assigned to, and managed by, virtual partitions. `dm-crypt` encrypts block devices with symmetric AES keys, called master-keys. Each encrypted LV contains a Linux Unified Key Setup (LUKS) [26] partition header, which is responsible for key management.

As a running Linux system requires some writable file system, the root "/" file system of the platform is assembled from multiple layers with an in-memory `tmpfs` to provide writable, but ephemeral storage.

Each application-specific logical volume contains one virtual partition application image, *i.e.*, a Terminal front-end which offers services to users.

Note that due to these structured file systems, complete measurements of the overall system configurations are composed of only a few hashes from well-known (read-only) software images and can therefore be compared easily with reference values in NFC-enabled mobile phones.

B. The NFC-Enabled TPM Prototype

In order to demonstrate an autonomic and NFC-compatible TPM that runs on the public terminal, we developed a low-cost NFC prototype. The prototype simply represents a TPM that is assembled on a Printed Circuit Board (PCB). Instead of conventional TPM modules, which are sealed and protected against modifications, we used an 8-bit microcontroller from Atmel (ATmega128) for our demonstration that can be freely programmed over a standard JTAG interface. The microcontroller has 128 kB of Flash memory, 4 kB of RAM, and operates at 13.56 MHz. Furthermore, the microcontroller is directly connected to an analog antenna circuit that has been also integrated on the PCB. It has been designed according to ISO/IEC 7810 and has a size of a conventional smart card (ID-1 format). This interface provides an easy access point for NFC-enabled devices. For debugging purposes, there also exists a serial interface on the PCB that allows a communication between the TPM prototype and a PC. In Figure 3, a picture of our NFC prototype is shown where it gets touched by an NFC-enabled mobile phone.

For RFID and NFC communication, our TPM prototype implements several protocol standards. It implements RFID protocols such as ISO/IEC 15693, ISO/IEC 14443 (type A),

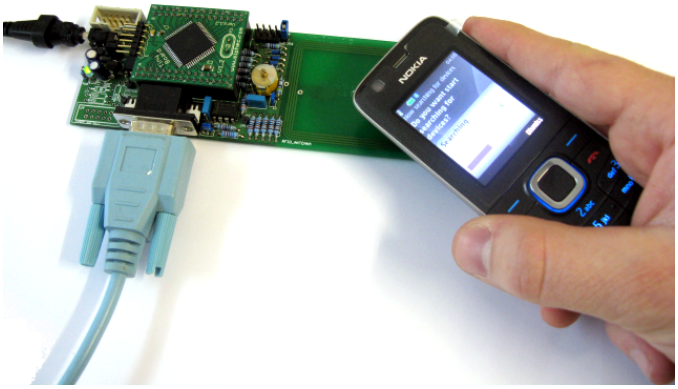


Figure 3. The NFC-enabled TPM prototype.

ISO/IEC 14443-4 and also ISO/IEC 18092. The software is written in C while parts have been implemented in Assembly language due to timing constraints. Moreover, it implements a user-command interface that allows easy administration over the serial interface. For our experiments, we have used the ISO/IEC 14443-A [19] protocol standard up to layer 4 using ISO/IEC 7816-4 Application Protocol Data Units (APDU) according to the NFC Forum type 4 tag operation specification [31].

In order to transmit and receive data from and to an NFC-enabled device, we encapsulated the payload using the specified NFC Data Exchange Format (NDEF). In particular, NDEF can be used by NFC-enabled mobile phones to allow an automatic launch of applications when the phone gets close to our prototype. For this, we support several NDEF records and allow user specific information to be transmitted to a mobile phone that get close to our prototype.

In order to sign the PCRs of the TPM prototype, we implemented ECDSA according to the recommendations of the National Institute of Standards and Technology (NIST). The implementation is based on the digital-signature standard [32] and uses the smallest recommended elliptic curve that is 192 bits for prime-field arithmetic. As a point-multiplication method (see line 2 in Algorithm 1), we decided to implement the improved Montgomery ladder proposed by Izu, Möller, and Takagi [22]. This method is shown in Algorithm 3 and performs a point addition and doubling operation in every Montgomery-ladder loop iteration to multiply the ephemeral key k with the fixed base point P of the elliptic curve. We performed all computations with (homogeneous) projective coordinates (the formulae used are given in [14]) and applied a coordinate randomization technique according to Coron [9]. For this, we randomized every intermediate value during the scalar multiplication

using a random value λ (line 1 in Algorithm 3). This makes our implementation more resistant to side-channel attacks [28] which try to reveal the ephemeral key used during the signature generation.

Algorithm 3 Montgomery ladder according to [22].

Require: $P = (P_x, P_y) \in E(\mathbb{F}_{p192}), k \in [1, 2^{192} - 1]$, random λ, $k \geq 2^{190}$.
Ensure: $Q = kP$, where $Q = (x, y) \in E(\mathbb{F}_{p192})$
1: $Q_0 = (X_0, Z_0) \leftarrow (\lambda P_x, \lambda)$.
2: $Q_1 = (X_1, Z_1) \leftarrow Dbl(P)$.
3: **for** $i = 190$ **downto** 0 **do**
4: $\quad (Q_{k_i \otimes 1}, Q_{k_i}) \leftarrow ECCAddDbl(Q_{k_i}, Q_{k_i \otimes 1})$
5: **end for**
6: $x \leftarrow x(Q_0) = X_0 \cdot Z_0^{-1} \pmod p$.
7: **Return** (x).

As shown in Algorithm 3, we combined both group operations which are point addition and point doubling to one operation ($ECCAddDbl$). The entire operation needs four variables to store the two projective curve points (X_0, Z_0, X_1, Z_1) and seven intermediate variables to maintain the intermediate coordinates. Note that no y-coordinates have to be maintained during point multiplication due to the use of the Montgomery ladder. In addition, for digitally signing with ECDSA it is not necessary to recover the y-coordinate because only the final x-coordinate is used after the scalar multiplication. All finite-field operations have been implemented in C except the modular multiplication algorithm which was implemented in Assembly language due to performance reasons. The multiplication algorithm uses a product scanning form (Comba multiplication) and applies the fast NIST reduction algorithm to reduce the result [13]. For modular inversion, we implemented the binary algorithm which has been adapted from the extended Greatest Common Divisor (GCD) algorithm from [13], [42]. As a modular reduction method, we applied the algorithm proposed by Barrett [5].

C. The NFC-Attestation Scenario

In the NFC attestation scenario, an NFC-enabled device is used to touch the antenna of the TPM prototype. For this scenario, we used the NFC edition of the Nokia 6212 mobile phone. It is shipped with an integrated RFID-reader chip that allows touching of NFC-enabled objects in the near proximity. Using the Nokia Software Developer Kid (SDK), we implemented a Java J2ME Midlet that runs on the phone. We implemented three threads: `SearchThread`, `SignatureGeneratingThread`, and `SignatureVerifyingThread`. The `SearchThread` handles the detection of passive NFC devices in the field. If our NFC prototype gets touched by the phone, it is detected by the `DiscoveryManager` and an `ISO14443Connection` is established by the Midlet. After that, the Midlet sends an ISO 7816-4 APDU to the prototype to start the attestation process. The prototype signs the PCR values together with the nonce N_0. For this, we implemented the same algorithm of ECDSA in a Java Midlet allowing signing and also verifying of digital signatures. The used ECC parameters such as the type of elliptic curve, the curve parameters (a and b), or the base point P have been fixed for both devices. After signing, the NFC-enabled TPM responds with the generated quote, *i.e.*, the Quote PCR values and their digital signature Sig_{AIK}. The mobile phone compares the received PCR values with reference values and verifies the signature using the public key of the AIK. Note that the phone also verifies the public-key certificate of the AIK that was signed by a PrivacyCA. This certificate can be transmitted over the air interface or can be installed together with the application Midlet that is used to perform the attestation with public terminals. A screen-shot of the mobile phone application is shown in Figure 4.

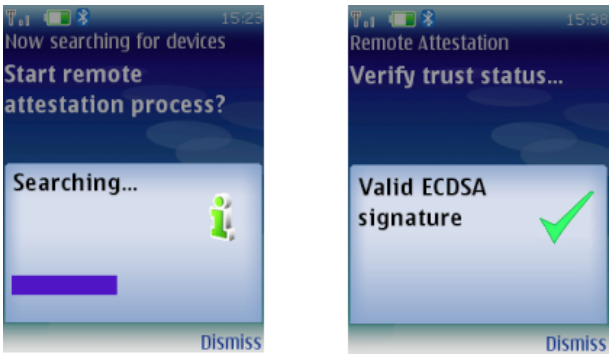


Figure 4. Screenshots of the remote-attestation procedure.

The implemented Midlet is composed of 46 Java files where 20 files are used to implement the cryptographic primitive of ECDSA. 26 files have been implemented for the certificate handling, the ISO144434 connection, and the user interface. The final executable *jar* file has a file size of 122 kB.

D. Performance

The digital-signature generation of our TPM prototype takes about 31 million clock cycles. Due to the characteristics of our scenario it is sufficient to consider only the performance of a single session. Running at 13.56 MHz this results in a running time of about 2.31 seconds to generate the signature. The verification of the signed message on the mobile phone takes 33 ms. The transmission time over the air interface has been measured using an 8-bit digital oscilloscope. The time between the first bit transmitted and the last bit received has been taken. First, we measured the anti-collision and initialization phase of the NFC protocol which

needs about 22 ms in our experiments. Second, the challenge N_0 and the Quote response are transmitted. For this, we assumed a typical number of different PCR values, *i.e.*, 6 in our experiments, resulting in 160+6*160+192=1 312 bits. The transmission takes about 140 ms (using a fixed RFID/NFC data rate of 106 kbit/s). Thus, the entire attestation process can be performed within three seconds. Note that we focused on a proof-of-concept realization that provides a practical demonstration of the proposed system architecture. Instead of a hardware implementation of the protocol, we implemented all routines in software. Existing TPMs already include cryptographic services such as RSA, where much more bits would have to be transmitted in contrast to elliptic curve implementations. As a comparison, the transmission of a 1 024-bit RSA-based signature (comparable with a 160-bit ECC implementation) would need 2 192 bits and roughly 240 ms transmission time which is almost twice as high as compared to the elliptic-curve based attestation protocol. This motivated our design decision, as we desire to keep the time the user needs to touch the public terminal as short as possible.

Table I shows the results of our ECC software prototype implementations. The first four lines show the code size and memory requirements of the implemented Assembler routines. Addition and subtraction need the same amount of resources which are 70 bytes of program code. Multiplication has been implemented to support different data widths and does not unroll the instruction sequences which results in 188 bytes of code. The reduction routine needs 752 bytes of code and has been optimized for the NIST reduction for $\mathbb{F}_{p192}$. The rest of the implemented files have been implemented in C. `ECC_Param.h` stores the needed elliptic curve parameters in the program code and needs 168 bytes. `ECC_FFOps.c` implements all necessary modular operations which invoke the Assembler routines. It needs 2 342 bytes of code. `ECC_PointMul.c` implements the ECC group operation that are addition and doubling which need 2 012 bytes of code. `ECC_Utils.c` and `ECDSA.c` implement utility functions such as array copying and the main loop of the scalar multiplication. Note that the implementation can be further optimized, *e.g.*, implementing all modular operations in Assembler, combining addition and doubling to reduce code size, or minimize function calls to reduce stack allocation. In total, our implementation needs 6 318 bytes of code and about 500 bytes of RAM memory. It is therefore suitable for integration in TPM circuitry.

Table I
IMPLEMENTATION RESULTS OF OUR NFC-ENABLED TPM PROTOTYPE.

File	Code [bytes]	Data [bytes]
ASM_ADD	70	0
ASM_SUB	70	0
ASM_MUL	188	0
ASM_RED	752	0
ECC_Param.h	168	0
ECC_FFOps.c	2 342	121
ECC_PointMul.c	2 012	218
ECC_Utils.c	84	50
ECDSA.c	632	98

VI. CONCLUSION

In this article, we proposed a new architecture that enables NFC capabilities to TPM devices. The architecture combines an NFC front-end and existing TPM functionalities into one piece of silicon and provides two additional connections for an RFID antenna. This approach allows users to simply touch NFC-enabled TPM devices with their mobile phones to verify the configuration state of a public terminal, for instance. We implemented a Midlet for the NFC-enabled Nokia 6112 mobile phone which makes a trust decision by applying the trusted computing primitive of remote attestation. The configuration states of a terminal gets digitally signed and the user gets informed on the display. Next to the mobile phone, we implemented a proof-of-concept NFC prototype that shows the practical realizability of our architecture. We implemented ECDSA on both devices and give performance results for a trust decision. We also modified the terminal software architecture to ease measurement and analysis of trusted states.

The outcomes of our work are as follows. First, it shows that trusted computing in NFC environments can effectively help to overcome confidentiality issues before the establishment of a potential distrusted terminal session. The primitive of remote attestation supported by common TPM modules can be used to provide a trust decision to users who want to establish a connection to a public terminal. Second, the rapid growth and widespread adoption of NFC in current hand-held devices like smart phones emphasize our decision for an integration of NFC into future TPMs. There already exist infrastructures for our proposed architecture such as the public-key infrastructure according to X.509 or the integration of ECC-capabilities in the Java framework. The integration of NFC into TPMs will pave the way for a ”touch’n trust” solution in upcoming applications.

ACKNOWLEDGMENT

The work has been supported by the Austrian government founded projects *PIT*, grant no. 825743, and *acTvSM*, grant no. 820848.

REFERENCES

[1] M. Hutter and R. Toegl, “A Trusted Patform Module for Near Field Communication,” in *Proceedings of the Fifth International Conference on Systems and Networks Communications*. IEEE Computer Society, 2010, pp. 136–141.

[2] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," in *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*. San Jose, California, USA: ACM, 2006, pp. 2–13.

[3] Advanced Micro Devices, *AMD64 Virtualization: Secure Virtual Machine Architecture Reference Manual*, May 2005.

[4] American National Standards Institute (ANSI), "AMERICAN NATIONAL STANDARD X9.62-2005. Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)," 2005.

[5] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in *Proceedings on Advances in Cryptology—CRYPTO '86*. London, UK: Springer-Verlag, 1986, pp. 311–323.

[6] F. Bellard, "Qemu, a fast and portable dynamic translator," in *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 41–41.

[7] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM conference on Computer and communications security*. Washington DC, USA: ACM, 2004, pp. 132–145.

[8] G. Coker, J. Guttman, P. Loscocco, J. Sheehy, and B. Sniffen, "Attestation: Evidence and trust," in *Lecture Notes in Computer Science*, vol. 5308/2008. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–18.

[9] J.-S. Coron, "Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems," in *Cryptographic Hardware and Embedded Systems – CHES'99, First International Workshop, Worcester, MA, USA, August 12-13, 1999, Proceedings*, ser. LNCS, Ç. K. Koç and C. Paar, Eds., vol. 1717. Springer, 1999, pp. 292–302.

[10] Free Software Foundation, "GNU Grub," 2010. Available: http://www.gnu.org/software/grub/ [accessed online June 11, 2011]

[11] S. Garriss, R. Cáceres, S. Berger, R. Sailer, L. van Doorn, and X. Zhang, "Trustworthy and personalized computing on public kiosks," in *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2008, pp. 199–210.

[12] D. Grawrock, *Dynamics of a Trusted Platform: A Building Block Approach*. Intel Press, February 2009, iSBN 978-1934053171.

[13] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Berlin / Heidelberg, Springer, 2004.

[14] M. Hutter, M. Joye, and Y. Sierra, "Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation," in *Progress in Cryptology - AFRICACRYPT 2011 Fourth International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011*, ser. LNCS, A. Nitaj and D. Pointcheval, Eds., vol. 6737. Springer, 2011, pp. 170–187.

[15] D. Safford, J. Kravitz, and L. v. Doorn, "Take control of tcpa," *Linux Journal*, vol. 2003, no. 112, p. 2, 2003. Available: http://portal.acm.org/citation.cfm?id=860399 [accessed online June 11, 2011]

[16] IEEE, "IEEE Standard 1363a-2004: IEEE Standard Specifications for Public-Key Cryptography, Amendment 1: Additional Techniques,", September 2004. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=9276 [accessed online June 11, 2011]

[17] Intel Corporation, "Trusted Boot," 2008. Available: http://sourceforge.net/projects/tboot/ [accessed online June 11, 2011]

[18] Intel Corporation, "Intel Trusted Execution Technology Software Development Guide," March 2011. Available: http://download.intel.com/technology/security/downloads/315168.pdf [accessed online June 11, 2011]

[19] International Organization for Standardization (ISO), "ISO/IEC 14443: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards," 2000.

[20] International Organisation for Standardization (ISO), "ISO/IEC 14888-3: Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms," 2006.

[21] International Organization for Standardization (ISO), "ISO/IEC 9594-8:2008: Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks," 2008.

[22] T. Izu, B. Möller, and T. Takagi, "Improved Elliptic Curve Multiplication Methods Resistant against Side Channel Attacks," in *INDOCRYPT*, ser. LNCS, A. Menezes and P. Sarkar, Eds., vol. 2551. Springer, 2002, pp. 296–313.

[23] A. Kivity, V. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux Virtual Machine Monitor," in *OLS2007: Proceedings of the Linux Symposium*, 2007, pp. 225–230.

[24] KVM Project, "KVM - Kernel-based Virtualization Machine," 2006. Available: http://www.linux-kvm.org/ [accessed online June 11, 2011]

[25] Q. Li, X. Zhang, J.-P. Seifert, and H. Zhong, "Secure Mobile Payment via Trusted Computing," in *Asia-Pacific Trusted Infrastructure Technologies – APTC, Third International Conference, October 14 - 17, 2008, Wuhan, China, Proceedings*. Hubei: IEEE, November 2008, pp. 98–112. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4683087&tag=1 [accessed online June 11, 2011]

[26] C. Fruhwirth, "New methods in hard disk encryption," Institute for Computer Languages, Theory and Logic Group, Vienna University of Technology, Tech. Rep., 2005. Available: http://clemens.endorphin.org/publications [accessed online June 11, 2011]

[27] LVM project, "LVM2," 2010. Available: http://sources.redhat.com/lvm2/ [accessed online June 11, 2011]

[28] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007, iSBN 978-0-387-30857-9. Available: http://www.dpabook.org [accessed online June 11, 2011]

[29] J. M. McCune, A. Perrig, A. Seshadri, and L. van Doorn, "Turtles all the way down: Research challenges in user-based attestation," in *Proceedings of the Workshop on Hot Topics in Security (HotSec)*, August 2007. Available: http://www.truststc.org/pubs/286.html [accessed online June 11, 2011]

[30] D. Molnar, A. Soppera, and D. Wagner, "Privacy for RFID Through Trusted Computing," in *ACM Workshop On Privacy In The Electronic Society, WPES, Alexandria, Virginia, USA, November, 2005, Proceedings*. ACM Press, November 2005, pp. 31–34. Available: http://www.cs.berkeley.edu/~dmolnar/papers/wpes05-camera.pdf [accessed online June 11, 2011]

[31] NFC Forum, "NFC Forum Type 4 Tag Operation - Technical Specification," NFC Forum, March 2007.

[32] National Institute of Standards and Technology (NIST), "FIPS-186-2: Digital Signature Standard (DSS)," January 2000, Available: http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf [accessed online June 11, 2011]

[33] B. Parno, "Bootstrapping trust in a "trusted" platform," in *Proceedings of the 3rd conference on Hot topics in security*. San Jose, CA: USENIX Association, 2008, pp. 1–6.

[34] M. Pirker, R. Toegl, D. Hein, and P. Danner, "A PrivacyCA for anonymity and trust," in *Trust '09: Proceedings of the 2nd International Conference on Trusted Computing*, ser. LNCS, L. Chen, C. J. Mitchell, and M. Andrew, Eds., vol. 5471. Springer Berlin / Heidelberg, 2009.

[35] M. Pirker and R. Toegl, "Towards a virtual trusted platform," *Journal of Universal Computer Science*, vol. 16, no. 4, pp. 531–542, 2010.

[36] M. Pirker, R. Toegl, and M. Gissing, "Dynamic enforcement of platform integrity (a short paper)," in *Trust '10: Proceedings of the 3rd International Conference on Trust and Trustworthy Computing*, ser. LNCS, A. Acquisti, S. W. Smith, and A.-R. Sadeghi, Eds., vol. 6101. Springer Berlin / Heidelberg, 2010.

[37] M. Pirker, R. Toegl, T. Winkler, and T. Vejda, "Trusted computing for the Java™ platform," 2009. Available: http://trustedjava.sourceforge.net/ [accessed online June 11, 2011]

[38] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Commun. ACM*, vol. 17, no. 7, pp. 412–421, 1974.

[39] M. Prokop et al., "Grml - debian based linux live system for sysadmins / texttool-users," 2010. Available: http://grml.org/ [accessed online June 11, 2011]

[40] A.-R. Sadeghi and C. Stüble, "Property-based attestation for computing platforms: caring about properties, not mechanisms," in *NSPW*, C. Hempelmann and V. Raskin, Eds. ACM, 2004, pp. 67–77.

[41] Software in the Public Interest, Inc., "Debian gnu/linux 5.0," 2010. Available: http://www.debian.org/releases/lenny [accessed online June 11, 2011]

[42] J. Stein, "Computational Problems Associated with Racah Algebra," *Journal of Computational Physics*, pp. 397–405, 1967.

[43] C. Strachey, "Time sharing in large, fast computers," in *IFIP Congress*, 1959.

[44] Trusted Computing Group, "TCG infrastructure specifications." Available: https://www.trustedcomputinggroup.org [accessed online June 11, 2011]

[45] Trusted Computing Group, "TCG TPM specification version 1.2 revision 103," 2007. Available: https://www.trustedcomputinggroup.org [accessed online June 11, 2011]

[46] Trusted Computing Group, "TCG software stack specification, version 1.2 errata a," 2007. Available: https://www.trustedcomputinggroup.org/ [accessed online June 11, 2011]

[47] R. Toegl, "Tagging the turtle: Local attestation for kiosk computing," in *Advances in Information Security and Assurance*, ser. LNCS, J. H. Park, H.-H. Chen, M. Atiquzzaman, C. Lee, T. hoon Kim, and S.-S. Yeo, Eds., vol. 5576. Springer Berlin / Heidelberg, 2009, pp. 60–69.

[48] R. Toegl and M. Hutter, "An approach to introducing locality in remote attestation using near field communications," *The Journal of Supercomputing*, 2011. Available: http://dx.doi.org/10.1007/s11227-010-0407-1 [accessed online June 11, 2011]

[49] R. Toegl, M. Pirker, M. Gissing "acTvSM: A dynamic virtualization platform for enforcement of application integrity," *Proceedings of INTRUST 2010: The Second International Conference on Trusted Systems*, ser. LNCS, Springer Berlin / Heidelberg, 2011. In print.

www.iariajournals.org

International Journal On Advances in Intelligent Systems
ICAS, ACHI, ICCGI, UBICOMM, ADVCOMP, CENTRIC, GEOProcessing, SEMAPRO, BIOSYSCOM, BIOINFO, BIOTECHNO, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS, CLOUD COMPUTING, COMPUTATION TOOLS
issn: 1942-2679

International Journal On Advances in Internet Technology
ICDS, ICIW, CTRQ, UBICOMM, ICSNC, AFIN, INTERNET, AP2PS, EMERGING
issn: 1942-2652

International Journal On Advances in Life Sciences
eTELEMED, eKNOW, eL&mL, BIODIV, BIOENVIRONMENT, BIOGREEN, BIOSYSCOM, BIOINFO, BIOTECHNO
issn: 1942-2660

International Journal On Advances in Networks and Services
ICN, ICNS, ICIW, ICWMC, SENSORCOMM, MESH, CENTRIC, MMEDIA, SERVICE COMPUTATION
issn: 1942-2644

International Journal On Advances in Security
ICQNM, SECURWARE, MESH, DEPEND, INTERNET, CYBERLAWS
issn: 1942-2636

International Journal On Advances in Software
ICSEA, ICCGI, ADVCOMP, GEOProcessing, DBKDA, INTENSIVE, VALID, SIMUL, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS, CLOUD COMPUTING, COMPUTATION TOOLS
issn: 1942-2628

International Journal On Advances in Systems and Measurements
ICQNM, ICONS, ICIMP, SENSORCOMM, CENICS, VALID, SIMUL
issn: 1942-261x

International Journal On Advances in Telecommunications
AICT, ICDT, ICWMC, ICSNC, CTRQ, SPACOMM, MMEDIA
issn: 1942-2601