Network Architectures
and Services
NET 2013-10-1

Dissertation

Autonomous and Robust Components for Security in Network Domains

Holger Kinkelin

Network Architectures and Services
Department of Computer Science
Technische Universität München

TECHNISCHE UNIVERSITÄT MÜNCHEN
Institut für Informatik
Lehrstuhl für Netzarchitekturen und Netzdienste

Autonomous and Robust Components for Security in Network Domains

Holger Kinkelin

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. rer. nat. Florian Matthes
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Georg Carle
	2.	Univ.-Prof. Dr.-Ing. Günter Schäfer (Technische Universität Ilmenau)

Die Dissertation wurde am 31.07.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 09.10.2013 angenommen.

Cataloging-in-Publication Data
Holger Kinkelin
Autonomous and Robust Components for Security in Network Domains
Dissertation, October 2013
Network Architectures and Services, Department of Computer Science
Technische Universität München

ISBN 3-937201-38-6
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)
DOI 10.2313/NET-2013-10-1
Network Architectures and Services NET-2013-10-1
Series Editor: Georg Carle, Technische Universität München, Germany

Abstract:

Computer networks and networked services are essential enabling technologies of many aspects of modern societies today. One necessary requirement to guarantee their correct functioning and availability is to ensure network security. In managed environments, such as in enterprise networks, trained professionals administer a wide range of security mechanisms. However, in these days also a large amount of networks without professional administrators exist, e.g., in private homes. Home networks of smart homes evolved into highly complex systems that include various, partially safety-relevant services. As such networks have high demands on security, but are operated by users with often only low technical skills, security support that does not need professional administrators becomes highly desirable.

The first part of this thesis focuses on the research question of how suitable access control can be integrated into home networks. As an answer a system for guided security management of network domains is presented. One component of the system allows using keying material certified by a local Certificate Authority as basis of identification and authentication. As there is a demand to interconnect network domains, e.g., to share services across smart homes, additional components of the system allow establishing strong and reliable trust relationships between network domains. This is complemented by a component to manage access rights in and between network domains.

Subsequently, requirements considering resilience and security of the platform used to host the security management system are considered. As home networks are no safe place, the threats of hardware failures and of malware that might extract cryptographic keys or that could modify the behavior of components is considered. The central question studied in the second part of the thesis is how an execution environment can be designed that satisfies the needs of a security management system in home networks. An architecture with virtualization as basis of the execution environment, resilience mechanisms, and with hardware-based security components including Trusted Computing and smart card technology to protect keys and to give evidence about the server's integrity, is investigated.

While the target scenario of this thesis are home networks, the contributions aim to be suitable to other environments such as small offices, building networks, or enterprise networks, where low administrative effort for security is desirable.

Kurzfassung:

Vernetzte IT-Systeme sind heute eine gesellschaftlich wichtige Schlüsseltechnologie. Netzsicherheit ist eine der wichtigsten Grundlagen, um das korrekte Funktionieren und die Verfügbarkeit dieser Systeme zu garantieren. In verwalteten Netzen von z.B. Großunternehmen werden daher vielfältige Sicherheitstechnologien eingesetzt und von professionellen Administratoren betreut. Heute existiert aber zudem eine steigende Anzahl von Netzen, z.B. in Privatwohnungen, die nicht von Administratoren verwaltet werden. Heimnetze in sog. Smart Homes entwickeln sich zudem zu äußerst komplexen Systemen weiter, die teils sicherheitskritische (im Sinne von safety) Dienste beherbergen. Derartige Heimnetze haben somit hohe Sicherheitsanforderungen (im Sinne von security) werden aber von Nutzern betrieben die meist ein nur geringes technisches Verständnis mitbringen. Daher werden nun Netzsicherheitstechnologien benötigt, die nicht von Administratoren betreut werden müssen.

Im ersten Teil dieser Arbeit wird die Forschungsfrage untersucht wie Zugriffskontrolle in Heimnetze integriert werden kann. Als Antwort wird ein Assistenzsystem für die Verwaltung von Identitäten und Zugriffsrechten in Netzwerkdomänen vorgestellt. Eine Komponente des Systems ermöglicht die Verteilung von lokal zertifiziertem Schlüsselmaterial und bildet die Grundlage für Identifizierung und Authentifizierung. Zusätzlich besteht der Bedarf Dienste über Heimgrenzen hinweg zu teilen. Aus diesem Grund werden weitere Komponenten entwickelt, die den Aufbau starker und zuverlässiger Vertrauensbeziehungen zwischen Heimen ermöglichen. Ein weitere Komponente zur Verwaltung von Zugriffsrechten vervollständigt das System.

Anschließend werden Anforderungen an Sicherheit und Resilienz einer zur Bereitstellung des Assistenzsystems geeigneten Plattform untersucht. Da Heimnetze keine sicheren Umgebungen sind wird insbesondere die Gefahr von Hardware-Ausfällen und von Malware, die vertrauliche Schlüssel extrahiert bzw. das Verhalten der Sicherheitskomponenten beeinträchtigt, berücksichtigt. Die zentrale Frage des zweiten Teils der Arbeit ist dementsprechend wie eine Ausführungsumgebung geschaffen werden kann, die den Sicherheitsanforderungen des Assistenzsystems gerecht werden kann. Eine auf Virtualisierung basierende Architektur mit Resilienz-Diensten wird beschrieben. Zudem werden auf sicheren Hardware-Komponenten, wie dem Trusted Platform Module und Smart Cards, aufbauende Mechanismen zum Schutz von Schlüsselmaterial bzw. zur Gewährleistung der Integrität der Ausführungsumgebung entwickelt.

Obwohl das Heimnetz das Kernszenario dieser Arbeit ist sind deren Beiträge allgemeingültig und in allen Umgebungen einsetzbar in denen geringer Administrationsaufwand gewünscht ist, z.B. in Büronetzen, in Gebäudenetzen oder in Netzen größerer Firmen.

Acknowledgments:

This thesis would have not been possible without the support I received from many individuals.

First of all I would like to thank Prof. Dr.-Ing. Georg Carle for giving me the opportunity to join the Chair for Network Architectures and Services and for supervising the dissertation. I also would like to thank Prof. Dr.-Ing. Günther Schäfer for being my second assessor and Prof. Dr. rer. nat. Florian Matthes for chairing the examination committee.

Most parts of this thesis were elaborated in the context of the research projects AutHoNe (Autonomic Home Networks) and ANSII (Anomaly Detection and Embedded Security in Industrial Information Systems), both funded by the German Federal Ministry of Education and Research. I want to express my gratitude to colleagues, both from the chair and from external project partners, who were involved in these projects. In particular I want to thank Dr. rer. nat. Andreas Müller and Dr. rer. nat. Heiko Niedermayer for their cooperation concerning the access control framework. Furthermore, I want to thank Simon Stauber for his collaboration on the security framework and his help revising the draft version of this thesis. In addition, my thanks go to my former students who contributed to this work. Lastly, I want to thank all colleagues for making the chair to the great place it is.

A doctorate influences ones private life much and especially finishing the thesis and preparing for the examination are quite intense. I want to thank my partner Judith Reuß for her support in all those years and especially during the last months, her understanding that I did not have much time for her lately and for proofreading the text.

Finally, I want to thank my parents Traudel and Albrecht Kinkelin who supported me for my whole lifetime and believed in me.

Garching bei München, December 2013

Holger Kinkelin

Contents

List of Figures

List of Tables

Listings

Part I

Introduction

1. Introduction

Over the last decades computer networks have been revolutionizing the world. They evolved into an essential backbone technology that influences or even enables many aspects of modern societies today. Due to their importance, networked computer systems are often the target of attacks. It is for this reason that security is one of the most important requirements on computer networks and also one of the most relevant research areas of computer science.

A large variety of security technologies have been developed in order to achieve secure networks. One of the most basic but also most important security mechanisms is **access control** which prevents unauthorized subjects from accessing networks or networked services. Professional access control systems, such as Kerberos or RADIUS, as well as many other security related technologies, are basically only deployed in network environments of larger companies or organizations today. One important reason for this limitation is the complexity of these systems, which can only be handled by trained system administrators.

Besides professionally managed networks a vast amount of networks exist in smaller companies, such as handicraft businesses or small office/home office (SoHo) environments but also in private homes. In most of these cases no professional administrator is available. Hence, we denote such networks as **unmanaged networks**. As a result, professional security technologies cannot be deployed here as their configuration, maintenance, and daily operation would be too difficult for laymen.

The central example for unmanaged networks used in this thesis are home networks. This network type is an especially interesting research subject due to multiple reasons. 1) In an average home typically no expert is available, who is capable to take care of network security. 2) Home networks have been developing over the last decade to highly complex systems. Examples for networked technologies in homes include networked entertainment technologies, but also networked home control services that finally implement the so-called **smart home**. Smart home services might even be relevant for the safety of the home and its residents when elements such as the homes alarm system or even doors and windows, can be controlled over the network. 3) Even private computer and network users become increasingly aware of their security and privacy and therefore start distrusting and even refusing products and technologies that do not consider security well enough. 4) No serious concepts how future home networks can be secured and how security can be managed by laymen exist today. Examples that substantiate this observation is the lack of authentication/authorization in UPnP-based audio/video streaming solutions or WLAN access control, which often suffers from insecure (short) passwords.

The result of this short consideration is that the gap between the demand on security technologies targeted to the needs of modern, networked home environments and their residents and the amount of available solutions grows.

1.1 Objectives and Research Questions

Today, many computer networks exist that are not operated by professionals. It is for this reason why the security of unmanaged networks is often neglected and even fundamental security objectives, such as authentication and authorization of users, are not or only insufficiently solved. The first objective of this thesis is therefore to increase the security of unmanaged networks by **creating an access control system suitable for unmanaged environments**.

As the home networks are no safe place, the threat of malware compromising the access control system or hardware defects destroying its settings are quite high. Hence, the second objective of this thesis is to **create a platform able to securely and resiliently host the components of this system**.

Our claim is that based on already existing, standardized enterprise grade security technologies and carefully designed software components that automate or guide the users through difficult to understand and perform tasks a similar level of security (considering access control) can be achieved in unmanaged networks as in professionally managed networks. Hence, **this thesis will provide answers to the following questions:**

Q1	How to design a user-centric identification/authentication scheme for unmanaged networks based on strong cryptography?
Q2	How to empower inexperienced users to effortlessly and securely use this identification/authentication scheme?
Q3	How to empower inexperienced users to use a sophisticated authorization system?
Q4	How to design a device able to host security services in a home environment securely, resiliently and user-friendly?
Q5	How to protect and guarantee the integrity of this device?
Q6	How to protect keying material used by the identification/authentication scheme against theft and abuse?

Positioning and Goals

Scientific work on home networks can be roughly structured into three fields of research. The first field are empirical studies, such as [1, 2, 3, 4, 5]. These works try to understand properties of home networks and properties of home network users. An important aim of these works is to understand problems users have with today's technology and needs of users. These works typically stress the growing gap between demand and supply for management and security solutions targeted to the home environment as we do.

The second field of research is targeted to innovative home networking technologies. These works are focused on the problematic interoperability of devices and services caused by heterogeneity and create networked smart home control systems. Research is mostly performed by company research departments, e.g., of control4, Honeywell, HomeSeer, and especially Microsoft Research [6, 7]. It seems that companies see in the smart home a future market. However, it should be expected that these projects also address network security issues. But this seems to be a side issue at most.

The third field tries to bridge the gap between the first two fields of research by researching on management and security solutions for the home environment. The works presented in

[8, 9], for instance, analyze home network properties and describe best practices how access should be controlled in smart home environments. Other works, such as [10, 11, 12, 13, 14], describe approaches for security architectures but either do not focus on user friendliness or do not take social and ownership structures of home networks and expectations of users into account (please also refer to Section 7.7.3), which we think are both highly important factors.

This thesis can be assigned to the third group of research works. We analyze properties of home networks ourselves, build upon the findings of works from the first group and create our own access control system for home networks.

1.2 Contributions and Document Structure

The contributions of this thesis include 1) concepts and implementations of mechanisms that integrate various standardized, enterprise grade security technologies into an access control system targeted to unmanaged networks, 2) services that facilitate the management of this access control system in order to allow inexperienced users to use and benefit from this technology, and 3) the architecture of and Auxiliary Services for a platform suitable to host the access control system securely and resiliently.

Table 1.1 shows the relationship from above-named research questions, contributions, and key findings presented in subsequent chapters. Additionally, the table assigns a number to each contribution and provides information about the contribution and the methodology of its evaluation. Key findings and contributions are listed in greater detail at the end of each chapter. The numbering used in these sections corresponds to the numbering of contributions used in the table.

Question	Chapter	Contribution Number	Contribution Name	Model/Architecture	Software Description	Prototype	Discussion
Part II: Components for Access Control in Network Domains							
1	4	4.1 - 4.3	Identification/Authentication Scheme	•			•
2	5	5.1 - 5.2	Guided Entity Registration Service		•	•	•
2	6	6.1 - 6.3	Guided Trust Exchange Mechanisms		•	•	•
3	7	7.1	TLS Handshake Interception		•	•	•
3	7	7.2 - 7.3	Guided Policy Administration Service		•	•	•
Part III: Robustness of Security Components							
4	10	10.1	Domain Server Architecture	•			•
4	11	11.1	Automated Domain Management Service		•	•	•
4	11	11.2	Automated Resilience Service		•	•	•
5	12	12.1 - 12.4	Automated Integrity Verification Services		•	•	•
6	13	13.1 - 13.2	TPM-based Key Protection		•	•	•

Table 1.1: Overview of Contributions

This thesis is structured into four parts. I) Introduction, II) Components for Access Control in Network Domains, III) Robustness of Security Components, and IV) Conclusions.

Chapter 1 in **Part I** is the overall introduction to this thesis. The chapter motivates and presents the research questions of this thesis and provides additional information.

Part II is dedicated to the development of an access control system suitable for unmanaged networks (Q1 - Q3). **Chapter 2** introduces the problem area, namely unmanaged networks and analyzes the properties of such networks at the example of home networks. This chapter also defines the overall requirements on the access control system envisaged in this thesis. Background information about used or competing technologies is given in **Chapter 3**. **Chapter 4** discusses the design of an identification and authentication scheme for unmanaged networks (Q1). The initial approach to split the unmanaged network into individual *Domains* assigned to and managed by individual users and to assign a certificate authority to each Domain is introduced. Based on this approach a hierarchical *identification/authentication scheme* that reflects the structure of a home network is described. **Chapters 5** and **6** describe *assistance systems* that empower the mostly inexperienced users of unmanaged networks to use the described identification/authentication scheme (Q2). These systems assist users with 1) the distribution of certified identities to devices and services within their own Domain and 2) the federation of identities between Domains. **Chapter 7** presents how access control in unmanaged networks can be extended with an enterprise grade authorization system, namely XACML (Q3). The chapter describes 1) how existing services that support TLS-based authentication can be connected to XACML and 2) how users can be empowered to specify and create authorizations rules effortlessly and in a fault-proof manner using our *knowledge-based* concept. The single service components we have just outlined form the so-called *Guided Security Management System* for Domains.

Part III of this thesis is dedicated to the creation of a device called *Domain Server* that acts as a secure and convenient to use server for the Guided Security Management Systems of all Domains that belong to the local network (Q4 - Q6). **Chapter 8** introduces various problems concerning the secure and robust hosting of services in a home and gives an overall definition of requirements on the Domain Server. Various existing enterprise grade (security) technologies used as basis for the Domain Server are introduced in **Chapter 9**. **Chapter 10** describes the architecture of the Domain Server, which is based on the XEN hardware virtualization system, and explains how the security and infrastructural requirements of the Guided Security Management System can be fulfilled (Q4). On the basis of this architecture, *Auxiliary Services* are designed in **Chapter 11** that 1) facilitate the handling and management of the Guided Security Management System or of the Domain Server and 2) take care for the resilience of Domains in order to prevent against negative effects of data loss caused by hardware failures or other problems. **Chapter 12** describes the concept and implementation of a technology based on Trusted Computing and smart card technology that verifies the integrity of the Domain Server (Q5). This technology prevents that users trust a Domain Server that is insecure as it was compromised by malware or even executes non-authentic software components of the Guided Security Management System, which might render this system useless. That last **Chapter 13** discusses how the previously described access control system can be extended with Trusted Computing technology in order to prevent various attacks on keys such as key theft or abuse (Q6).

Chapter 14 in **Part IV** concludes this thesis by summarizing its contributions and describing future work.

1.3 Applicability

Even though most contributions are designed with regard to the home network, their applicability is not limited to this environment. All concepts can be transferred to other scenarios, as developed technologies are based on standardized technologies and therefore highly compatible to existing technologies in enterprise environments. Smaller companies with unmanaged networks can therefore benefit directly from the developed technologies. However, our technologies can also be used in managed company networks in order to disburden the network administrator.

1.4 Visual and Typographic Conventions

For the simplification of reading, following typographic conventions are used within this thesis:

emphasized	Newly introduced terms are emphasized where they are first mentioned.
italic	Quotes are set to italic.
`typewriter`	Commands, source code fragments, etc. are set in typewriter font.
bold	Visual highlighting uses bold font.

Requirements on a solution are enumerated alphabetically, e.g., Requirement RA, Requirement RB, etc. Sub-requirements of a requirement are enumerated numerically, e.g., Requirement RA.1, Requirement RA.2, etc.

Messages or processing steps in message sequence diagrams or actions in workflow diagrams are numbered. This number is used in the textual description of the protocol or workflow.

Tables that summarize findings or analyses, or compare findings to the state of the art use symbols to indicate the quality of a concept, solution, etc. The meaning of symbols is explained below:

✓	Fully applicable, good solution, positive properties, satisfies requirements fully, etc.
❍	Partially applicable, satisfying solution, mostly positive properties, requirements mostly fulfilled, etc.
✗	Not applicable, poor solution, problematic properties, requirements not fulfilled, etc.

Finally, this text will not use both the male and female forms of gender-specific pronouns (he/she, his/hers, etc.) for brevity and clarity, but only the female forms.

Part II

Components for Access Control in Network Domains

2. Introduction to the Problem Area

This chapter serves as an introduction to the problem area addressed in the present part of the thesis, namely access control in unmanaged networks, and acts as an additional motivation for this research work.

Chapter Structure

First our understanding of the term *unmanaged network* is clarified in Section 2.1. Next, home networks are analyzed as a specific example of unmanaged networks in Section 2.2 in order to understand which properties exist that influence the design of an access control system targeted to this environment. Section 2.3 analyzes shortcomings of currently used access control system in homes. At the end of this chapter overall requirements on the envisaged access control system for unmanaged networks are defined, see Section 2.4.

2.1 Unmanaged Networks

The term *unmanaged network* was already introduced in the overall motivation of this thesis. At this place a definition shall be given: A unmanaged network is understood in this thesis as a network that is not administered by professionals.

The lack of administration can be caused by several reasons. First, because no system administrator is available at all, which is in home networks, very small businesses, medical offices, etc. commonplace. Second, even in bigger companies parts of the network might exist that are not managed by a system administrator, think for instance of a (sub-) network of a department, a test bed, etc. In this case, possible overburdening of the system administrator or economic considerations might cause the lack of administration.

In both cases various problems might occur if access control is not solved properly. For this reason an access control system for unmanaged networks is envisaged in this thesis.

The focus of this work is clearly on home networks but results can be transferred to smaller and larger company networks. The developed technologies act here as a replacement of a missing administrator or disburden her.

2.2 Analysis of Home Networks

In the following, home networks are investigated in order to understand which properties they have, how home networks are used, and how home networks will develop in future.

The analysis is split into a technical part and a part where the user is in focus. This second part is especially important as most design criteria for an access control system for unmanaged networks will be influenced by the properties, expectations and needs of the users.

2.2.1 Technical Aspects

Over the last years the major purpose of a typical home network was to connect a small amount of networked devices to the Internet. For some years now, home networks started to change fundamentally. These changes were driven mostly by three factors:

- Dropping prices of computer hardware
- Dropping prices of broadband residential Internet connections
- The advent of home automation in average households

These factors have already changed the average network deployed in homes, and will continue to change the average home network even more.

Growing Amount of Networked Devices

Studies as the *Onlinestudie* [5] performed annually by the German public TV stations ARD and ZDF show that homes are populated by a large amount of networked devices already today. According to the study, most residents of a home possess a personal computer or notebook and additional devices, such as a smart phone or tablet computer. Another example for a networked device mentioned in the study is the "smart" TV set. According to the study, most of these devices connect to the home network using fast WLAN technologies.

In close future it must be expected that even more devices will be deployed in average homes that are able to connect to the local network. This is because "smart" devices with network interfaces will become mainstream. So it can be expected that most consumer electronics, such as digital cameras, video recorders and hi-fi systems, will be equipped with wireless or wired LAN interfaces in close future.

Additionally, the future will bring a growing amount of home appliances that are connected to the local network. Lighting [15], heating [16], kitchen equipment [17] and door and window controls [18] are first examples of smart home appliances that can be controlled via the local network. Finally, also networked sensor nodes that detect temperature, humidity, user presence, etc. will be deployed in average households in order to automate the home.

Growing Amount of Security and Safety Relevant Networked Services

In the past, almost no networked services were deployed in average homes. But already today, a small amount of services hosted within the average home can be found. Examples include audio/video (A/V) streaming services or storage and backup services hosted on consumer devices. In future the amount of networked services and especially their criticality will grow.

Home automation is the first reason for the growth of the amount of highly security relevant networked services in average homes. Currently, the manufacturers of networked home appliances only provide quite simple interfaces to their products, which can be accessed using a control application installed on a computer, smart phone or tablet computer, or

a web browser. Using such control "apps" it is for instance possible to modify the room temperature, switch on the lights or even open doors or windows.

The different home automation domains, e.g., heating, lighting or entertainment, are today still disconnected, as devices, applications and services of different manufacturers are not able to interact yet. This inability is caused by heterogeneity of communication protocols and media. But various research projects aim to bridge this gap. Microsoft Research, for instance, works on a concept similar to an operation system of a computer. The aim of the so-called *HomeOS* [6] platform is to offer uniform access to all devices and services within the home. For this reason it is fair to assume that in the near future intelligent home automation services will emerge that are able to control devices and home appliances across different vendor domains.

The second example supporting the claim that more security relevant services will emerge in homes is that average users become more and more aware of their privacy and begin to distrust services offered in the public Internet. A logical consequence is that users want to host services on own hardware in their homes in order to be sure that their valuable personal data valuable to them and possibly others is secure and neither abused by the service provider nor tapped by monitoring programs of secret services. Inexpensive and energy preserving hardware and fast broadband Internet accesses provide the basis for services hosted in homes. Various open source projects were started that provide needed software. Examples include *ownCloud* [19], which implements a synchronization service for files, calendars or contacts between private devices and *Diaspora* [20], which implements a distributed social network.

With both described trends, the unsolved issue of access control in home networks (see Section 7.7.3) finally became an important problem. The illegitimate access to control services of the smart home is critical for the safety of the home. Unauthorized access to services hosting data will harm the privacy of residents.

2.2.2 Non-Technical and Social Aspects

Besides the more technical aspects of home networking, which were just discussed, it is important to understand which needs and expectations the user of a home network has. For this reason, social aspects of home networking are investigated in the following.

Administrative and Ownership Structure of Home Networks

Most homes are inhabited by several residents. As a consequence, a mixture of private and public *entities* (devices and services) exist in home networks [2, 3]. *Private* entities are owned and used primarily by an individual person. Examples for private entities include a laptop, a smart phone or a storage service hosted on a personal device. Besides personal entities, the home network also contains *public* entities. Examples include devices and services collectively used by all residents, such as the services that control the smart home, a TV set, or an A/V streaming service. Another collectively used infrastructural service is the LAN/WLAN deployed in the home.

This ownership structure creates an entirely different situation compared to a company, where all entities connected to the corporate network belong to the *same* owner, i.e., the company. Therefore, devices and services inside a corporate network can and should be controlled by a central authority, i.e., by the administrator. In a home, typically *no* single authoritative user should exist that controls the home network and all connected entities. Instead, the management of entities should typically be distributed, i.e., be left to their respective owners. An exception might be that all or only some residents of a home decide that they do not want to be bothered with any kind of technical administration. Instead

they chose to allow, for instance, the person with the best technical skills in the home to administer their entities.

For this reason we regard a home network as a (virtual) structure that consists of several independent administrative regions. We call these regions *Network Domains* or simply *Domains*. Furthermore, it appears to be useful to differentiate between Domain types. A *Home Domain* is the logical structure that contains the network of the home and all public entities. Typically, the Home Domain is managed by the home owner or the technically most experienced resident of the home. *User Domains* belong to different residents of a home. Each User Domain contains the private entities of a specific resident and is typically managed by this person. Regardless of the Domain type, the authoritative user of a Domain is called *Domain Owner*.

Sharing of Resources

We and other research works [2, 3] expect that services hosted in home networks need to be shared between residents and that residents want to use their technology for collaboration. Examples include the control service of the smart home, which belongs to the Home Domain but needs to be accessible by all residents. Another example is sharing of data (photos, music, videos) between family members.

The sharing of data with family members, friends or colleagues is one of the most popular features of social networking services, such as Facebook, Google Plus, etc. For this reason we assume that data and services (subsequently subsumed with the term *resources*) need to be shared with persons that live *outside* the own home.

Furthermore we want to point out that persons that live in the same home have a special relationship with each other. It is for this reason that the likelihood will be very high that such persons want to share resources with each other. For this reason, a Home Domain can also be seen as a structure that contains the User Domains of all residents, i.e., there is a *belongs to* relationship between User Domains and Home Domains.

Access from Remote

Users will want to access resources present in a User Domain or the Home Domain they belong to from remote. The same is obviously true for persons a Domain Owner has shared resource with that live outside the home.

Growing Security Awareness of Users vs. Low Technical Expertise

As security is a feature that has no directly visible benefit for the user, users did not care too much about security in the past. Thanks to the media and the fact that networked technologies became quite familiar to ordinary users with time, the situation changed. Today, average users have a quite good understanding that security and privacy in home networks and the Internet is important for them. A study on tools for network management confirms that *"many"* of the participating persons *"were active in managing security and access control."* [1]

The results of this growing security awareness are clearly visible today. A decade ago, many open WLANs could be found in residential areas. Today only a very small fraction of WLANs are completely unprotected (no access control at all). Most users seem to understand that controlling access to their home network is important. Another example for the growing awareness of average users concerning security and privacy are social networks and online communities. The already mentioned Onlinestudie [4] showed that 85% of the users of social media services are aware of privacy issues and configure their online profiles in social platforms accordingly.

Unfortunately, average users still have not enough experience and background knowledge to deal with complex administrative tasks, e.g., managing access control in their home network. The already mentioned study on management tools confirms this statement: *"There were many non-expert users who had concerns related to access control, but could not actively enable or manage access control because of lack of knowledge about how to do so."* [1]

In the past, security aspects of products were often neglected by the industry. The manufacturers got away with missing and unthought security functions, as they knew that their customers did not care much about security. However, we are convinced that the growing security awareness of users will sooner or later have a high impact on their buying behavior. Users will not only rate products or services based on their functionality but also regarding security and privacy. Especially technologies related to the smart home, most probably the last truly private resort in the networked world, will be refused by users when these products do not implement security sufficiently. Therefore, research on user-friendly mechanisms for network security in homes is important.

Costs

Another important factor that must be taken into account are costs. Users still do *"not want to pay extra cost just for a management tool."* [1] The same will be true for other items that may cause substantial cost, e.g., hardware dedicated to the access control system. Time and effort needed for security management, which are both non monetary cost-factors, also need to be low.

2.2.3 Example

To give a practical example for the above-mentioned considerations, we want to present a simple home network that contains different Home and User Domains, see Figure 2.1. According to our considerations, this is a valid example how different users within a home or how different home networks might interact with each other.

In Home Network A (represented by the gray cloud) public services, for instance, an A/V streaming server and a home control service are located. The A/V server is accessed (service access is depicted in the figure by blue arrows) either by public devices (collectively used smart TV sets) or by private devices (personal notebook) that belong to a User Domain.

Typically, the home owner will choose to grant access to most public services to all residents of the home. Nevertheless, exceptions might exist from this rule, such as young children who are not allowed to access the home control service, as this service is critical for the homes safety.

Public and private services might also be accessed from the outside of the home. An example of this use case would be, as already mentioned, when the home owner grants access to the home control server to a neighbor (Home Domain C).

In contrast to public services, private services are typically only accessed by their owner. However, in the example a resident is allowed to access a service in a different User Domain (access between User Domains Z and Y). Finally users that belong to other home networks might be allowed to access resources in a local User Domain.

2.2.4 Synthesis

Home networks have been evolving into highly complex networked environments populated by numerous devices and services. In close future services will emerge that are critical for

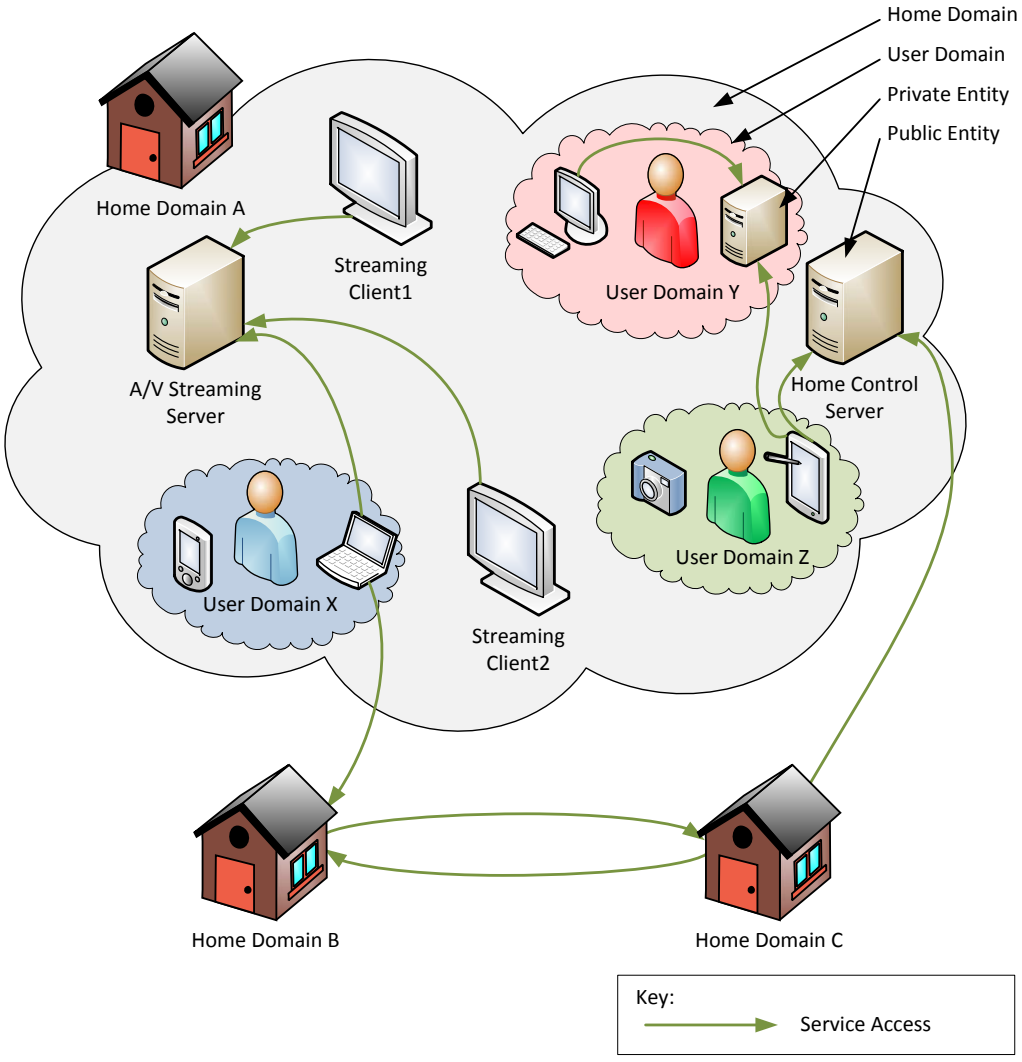


Figure 2.1: Example: Access to services located in Home and User Domains.

the home's safety and security. Users become aware of security and privacy problems but typically do not have enough experience to take care for their network's security. Therefore, security technologies targeted at this environment are urgently needed. Such a system needs to reflect the administrative and ownership structure of the home network and is easy to use.

2.3 Case Study: Access Control in Home Networks

Access control in home networks is an unsolved problem today. We want to substantiate this observation using two examples of widely deployed services in private homes, namely Wireless LANs and DLNA A/V streaming.

2.3.1 Wireless LAN

Wireless LANs (WLANs) are infrastructural services deployed in most home networks and one of the only examples of services used in private environments that use access control. Most WLAN Access Points (APs) implement the *WPA2 (Wi-Fi Protected Access)* standard for access control, which is regarded as cryptographically strong. Despite this fact, WLAN security is still improperly solved within unmanaged networks, which we want to explain subsequently.

In home environments WPA2 is typically operated in *Personal Mode*, which means that a device accessing the WLAN authenticates itself to the AP using a shared secret password. In most cases the user that maintains the WLAN AP selected this password. It is quite likely that this user selected a password that can be memorized well and that can be configured to an AP and other devices easily. Typically, such a password is short or composed of natural words, which make brute force or dictionary attacks feasible. Weak passwords must be regarded as a major threat for WPA2 protected WLANs.

The use of shared passwords causes further issues and impairs usability and flexibility of WLANs. For instance, the revocation of the right to access a WLAN from a specific user is a cumbersome process. Once a user has received the WLAN password, she is able to access the WLAN as long as the password is not reset. Resetting a WLAN password requires configuring a new password into the AP and all devices that should still be allowed to access the WLAN. The same issue prevents that the well known best practice to reset passwords now an then can be applied. Refreshing the WLAN password in short intervals will prevent that a leaked password can be used for a long time. Both factors, missing revocation of guest access and the inability to reset passwords, might negatively affect WLAN security on the long run.

An alternative to manual WLAN password configuration is *WPS (Wi-Fi Protected Setup)*, which is a home network technology standardized by the Wi-Fi alliance [21]. The purpose of WPS is the easy set-up of secure WLAN connections between Access Point (AP) and devices. Unfortunately, a major design flaw in PIN-based WPS was revealed in December 2011. This problem affects the security of WPS implementations of numerous devices by various manufacturers [22]. According to the vulnerability report published by the US-CERT [23], the manufacturers do not provide suitable firmware patches yet. US-CERT therefore advises to turn PIN-based WPS off.

To sum up, WLAN security of unmanaged networks can only be guaranteed today, when a user has sufficient expertise and is willing to configure WLAN access control manually. Inexperienced users are left alone with the security issues of their WLANs. So, in many networks users risk that their WLAN can be broken, which causes security, safety and even legal problems [24].

2.3.2 DLNA-Certified A/V Streaming and UPnP

The *Digital Living Network Alliance (DLNA)* [25] is an alliance of consumer electronics manufacturers and also the name of a family of standards for home entertainment. DLNA is a quite common technology in today's homes as most smart TV sets can be used as DLNA player that receives media from a central DLNA A/V streaming server.

The DLNA protocols are derived from the UPnP (Universal Plug and Play) standards [26], which cover service discovery and service control in a local network. A central problem of UPnP is the lack of authentication. The standards assume that all users and systems in the local network are allowed to access all services. This is not reasonable, think for instance of parental controls. Hence, services that require access control must implement this feature themselves.

DLNA-based entertainment solutions exist that include access control mechanisms for parental control. Here, access control is based on the idea to restrict access of devices identified by their IP- or MAC-addresses. A per-user authentication does not exist. Obviously, this is no sufficient solution.

2.3.3 Synthesis

This short consideration has shown that access control in home networks is unsolved today. Several common problems seem to exist:

- The lack of individual identities for users, devices, services, etc.
- The lack of authenticators that allow users, devices, services to claim their identity
- The lack of mechanisms that assist users in the deployment of identities in their network
- Security is still regarded as an option, not as a must.

2.4 Overall Requirements

Based on the analysis of properties of smart home networks and their users and the study on access control in home networks, the following overall requirements are defined on the access control system targeted to unmanaged networks. These overall requirements will be refined and detailed in following chapters.

Requirement A (RA): Secure Identification and Authentication Scheme:

Unmanaged networks, as future smart home networks, have a high demand on security. For this reason, access to resources needs to be controlled. A first step in this quest is to introduce a scheme for identification and authentication of entities. Furthermore, the authentication of entities must be based on strong (cryptographic) principles in order to be secure. This requirement will be addressed in Chapter 4.

Requirement B (RB): Flexible and Fine Grained Authorization:

Unmanaged networks are populated by a growing amount of public and private devices and security relevant services that offer various resources. Access control based on the mere authentication of entities is not flexible enough. For this reason, additional authorization mechanisms need to be deployed that control access flexibly and in a fine-grained fashion, see Chapter 7.

Requirement C (RC): Decentralized Management:

Depending on the type of unmanaged network it is required to enable individual users that belong to this network to manage access to their own devices and services individually. Especially in a home network, which consists of several individual Domains, a centralized management approach should not be used, as this approach would clearly contradict the user's expectations and needs.

Requirement D (RD): Access Control across Domains and Homes:

The access control system for unmanaged networks must be capable to deal with entities that belong to different Domains. These Domains might be part of the local home or be part of another home.

Requirement E (RE): User Friendliness and Security per Default:

Deploying an access control system to an unmanaged network will only be successful and possible, if this system is 1) deeply integrated and mostly transparent to the users and 2) manageable by non-expert users. For this reason, difficult to understand technical details need to be hidden from the user and the user needs to be guided through the security configuration of her personal Domain.

3. Background

After setting the context of our work in the previous chapter, we want to familiarize the reader in this chapter with the issue of access control and the vocabulary used in this thesis.

Chapter Structure

First, access control and the related concepts identification, authentication and authorization are defined and explained, see Section 3.1. After this rather theoretical introduction, Section 3.2 discusses different authentication factors. Finally, existing models to establish trust in an asymmetric key pair are investigated and their properties evaluated, see Section 3.3. The last Section 3.4 of this chapter introduces various enterprise grade access control and authorization systems.

3.1 Access Control

Computers or computer networks offer resources (data, services) that need to be protected against unauthorized access. This protection is achieved through *access control*.

Access control is defined in the Internet Security Glossary as *"a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities according to that policy"* [27].

To understand this definition we want to delve in further and give additional definitions, explanations and examples. Again, the definitions are quoted from the Internet Security Glossary.

An *entity* is described as *"an active part of a system - a person [... or] an automated process [...]"* [27]. According to this definition, human users and services are entities. In contrast to the Internet Security Glossary, we understand devices and services as entities. Human beings, which own entities, are called *users*.

A *security policy* is *"a set of policy rules [...] that direct how a system provides security services to protect sensitive and critical system resources"* [27]. An example of a security policy might be a rule that expresses which entity ("who") is authorized to access a resource ("what") for which purpose ("why") and under which circumstances (e.g. "when" or "from where").

Authorization finally is defined as *"an approval that is granted to a system entity to access a system resource"* [27].

For every access control system a representation of the entity on the accessed system is needed. This representation is called *identifier* or *identity*, see Figure 3.1. Furthermore, additional means are needed for the entity in order to convince the system that the identity being claimed is indeed hers. This process is known as *authentication*.

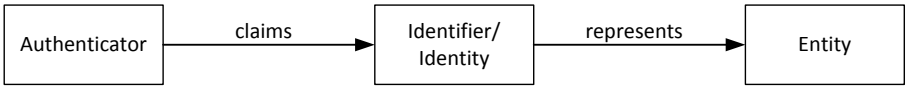


Figure 3.1: An identity/identifier is the representation of an entity in a system. An authenticator is used by an entity to claim her identity.

Identification is defined in the Internet Security Glossary as *"an act or process that presents an identifier [...] to a system so that the system can recognize a system entity and distinguish it from other entities"* [27]. Identifiers are generally considered as information that must not be kept secret. Examples include email addresses, account names, etc. After providing an identifier, the authentication step follows.

Authentication is described as *"an act or process in which an entity provides additional information (also authenticator) that must correspond exactly to the identity professed"* [28]. Unlike the identifier, the authenticator is considered as secret information, e.g., a secret password. For the sake of effectiveness and security of the access control system, it is inevitable that the authenticator remains strictly private. Otherwise, illegitimate entities might abuse the authenticator.

Finally please note that the term *authentication* is frequently used to refer to the combination of the identification and authentication process.

3.2 Authentication Factors

In order to claim an identity, different *authentication factors* can be used. According to the literature, e.g., [28], three different authentication factors are distinguished.

Knowledge

The *knowledge-based* identification/authentication scheme is the most widely used scheme to claim the identity of an entity in a system. Typically, the authentication is based on a simple challenge-response protocol that verifies if an entity *knows* the authenticator that belongs to an identifier. A typical example would be a valid username/password combination.

Possession

Possession-based authentication schemes are built around the proof that an entity *possesses* a specific authenticator. A cryptographic token that houses the private part of an asymmetric key pair is an example. An authentication protocol is used to prove that the authenticating entity possesses the private key that matches the public key, which is used as identifier. This so called *proof of possession* is typically based on the evaluation of a cryptographic signature computed using the private key.

Possession-based authentication can be implemented as a pure software solution as well. The private key (authenticator) then is stored on the hard drive of a computer and the cryptographic operation is processed in the CPU of the computer.

Biometry

Finally, the *biometry-based* authentication scheme exists. This scheme is built around the recognition of unique physical properties of a human being, such as fingerprints, retina patterns, or facial features. A special device, e.g., a fingerprint scanner, reads the biometric information and compares the input to stored reference information. Hence, the authentication is based on something the user *is*.

Multi-Factor Authentication

The system that authenticates an entity is not restricted to use a *single* authentication factor. In environments with high security needs, several authentication factors may be combined in order to successfully claim an identity. The so-called *multi-factor authentication* typically uses *different* authentication factors, e.g. a cryptographic token (possession) in combination with a PIN (knowledge). This approach strengthens the authentication quality, as if only one of the required authentication factors is missing, an adversary will not be able to authenticate herself to the system.

3.3 Trust Models

Cryptographic authentication protocols based on asymmetric cryptography prove that an entity possesses the private key (the authenticator) that belongs to a certain public key (the identifier). This proof is only meaningful for the authenticating system[1], when this system knows which public key belongs to which entity. In different words, the authenticating system must *trust* a public key, i.e., be confident that a certain public key belongs to a specific entity.

A *trust model* describes a way, in which an authenticating system that wants to use a public key as identifier for an entity can map this public key to an entity. Depending on the trust model used, the trustworthiness of the achieved key to entity mapping differs.

3.3.1 Direct Trust Model

The most simple trust model a system can use is to trust a public key as identifier of an entity only, when this key was *directly* received from this entity.

This simple approach is fully functional but can only be used in smaller groups, because exchanging and managing public keys manually is time consuming and complex. For this reason, the direct trust model does not scale well to larger groups. Nevertheless, the direct trust model offers the highest trustworthiness, as the public key to entity mapping does not depend on any third parties.

In order to address the mentioned usability issue, *Public Key Infrastructures (PKI)* were created that simplify the trust establishment into a public key. Two approaches, namely the centralized X.509 and the decentralized PGP/GPG Web-of-Trust, are discussed in the following.

3.3.2 The Centralized X.509 Public Key Infrastructure

The first way to build a highly scalable trust model are *centralized* PKIs. Today, the most widely used centralized PKI is the X.509 PKI [29]. X.509 is typically used in the context of web site (secure HTTP, HTTPS) or email (S-MIME) security.

[1] The system that authenticates an entity.

Approach

X.509 is built around the idea to have a central *Trusted Third Party (TTP)*, comparable to a notary, that issues signed *public key certificates*. A public key certificate, also called *identity certificate* or simply *certificate*, is a data structure that maps a public key to an entity or user. More precisely, the public key (which itself is an identifier) is mapped to another identifier of the user or entity. This identifier is typically human readable, e.g., an email address, the URL of a website, etc. In order to make the certificate verifiable, it is signed by the private key of the TTP, which is typically called *Certificate Authority (CA)*.

Before issuing a certificate for a public key, a CA needs to validate the identity of the entity or user that owns this key. This needs to be done with high accuracy, for instance by verifying the key owner's identity (i.e. her name) using a passport or national identity card. This non-technical process is crucial for the trustworthiness of the certificate.

A system that later wants to use a certificate must be convinced that the CA, which has signed this certificate, performed her duties accurately. In other words, this system must *trust* this CA in order to trust certificates issued by it. Besides trusting a CA, the system must also be able to verify the signatures of this CA. For this purpose the system needs the CA's public key and trust this key. In case the CA is *private*, for instance, a corporate CA, the system administrator can simply install the CA's public key on all devices that belong to the company.

In case a CA is *public*, for instance a commercial CA that sells certificates for web sites, things become more difficult. To solve this issue, manufacturers of software, such as web browsers, email clients or operation systems, typically ship their software with an selection of public keys that belong to well-known CAs considered to be trustworthy.

CA Hierarchies

As the amount of public CAs is vast, it is almost impossible to possess the public key of every existing CA. The answer to this problem is given in the *hierarchic* nature of the X.509 PKI, see Figure 3.2. CAs are able to issue certificates to other CAs in order to make them to a valid sub CA.

A system that wants to validate the certificate of an entity issued by a sub CA can easily establish trust into this certificate by validating the complete certificate chain from a well-known *root CA* to sub CA to the entity's certificate.

Such certificate hierarchies are not limited to only three layers. Multiple *intermediate CAs* might exist between a root CA and the CA that issued a certificate. For this reason, trusting a quite small amount of root CAs is sufficient to establish trust in certificates issued by a large amount of intermediate and sub CAs.

Cross-Signing

A different approach to establish trust in another CA is *cross-signing*. Cross-signing denotes the pairwise issuing of a certificate for the public key of the other CA. A cross-signed certificate expresses that the certified CA is also trustworthy.

This approach does not scale well, as the amount of needed cross signed certificates grows almost by the square of the number of participating CAs. Given n CAs, $n * (n - 1)$ cross-signed certificates are needed. For this reason, cross-signing is no alternative but a complementary concept to a hierarchical PKI, which is useful when no common root or intermediate CA is available.

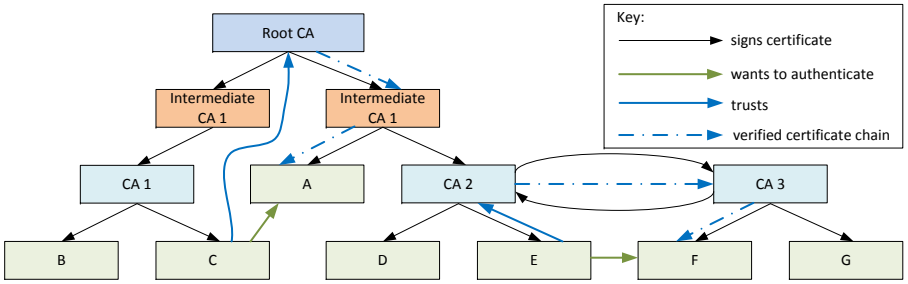


Figure 3.2: The X.509 Public Key Infrastructure: Entity C is able to establish trust in entity A's public key as a chain of trust can be established to A's certificate originating from the trusted root CA. Due to the cross signing of CA 2 and CA 3, entity E and F may authenticate each other as well.

Drawbacks

The major problem of the X.509 PKI is that it is almost impossible to decide if a trustworthy CA signed a certificate. The problem is aggravated the longer the certificate chain presented by an entity becomes. With each hierarchy layer added, the chance is increased that a CA is present that does not verify identities well enough before it issues a certificate, that a CA in the chain is compromised and issues faked certificates, or even that a CA is deceptive, i.e., intentionally signs faked certificates.

Most users are not aware that this problem exists at all. This is because the decision whether to trust a CA or not is typically shifted to the software manufacturers. The only option the average user has is to hope that decisions made by the software manufacturers were wise. For this reason, public X.509 PKIs might even be seen as an incapacitation of the user.

Benefits

Despite the drawbacks of X.509, there are many advantages. The major advantage of X.509 is that a large amount of certificates can be verified automatically when a trusted root CA exists. This is beneficial when a well known-party, e.g., an HTTPS-enabled web server of a bank, needs to be authenticated to other entities, e.g., the web browsers of this bank's customers.

CAs are also especially useful for closed groups that all trust the same private CA, for instance, a company CA. The reason for this is that the public key of the company CA can be directly deployed to the devices used by the employees. Additionally, the CA is controlled by the company and can be considered as trustworthy in this case.

3.3.3 The Decentralized PGP/GPG Web-of-Trust

A different approach to build a highly scalable PKI is the so called *Web-of-Trust (WoT)*. This trust model is used in Pretty Good Privacy (PGP), GnuPG (GPG) or other implementations that follow the OpenPGP standard [30]. PGP/GPG are typically used in the context of email security.

Instead of being centralized as X.509, the PGP/GPG WoT is user-centric and self-organized. The users of this system map another user's identity to this user's public key using certificate-like data structures signed by their own private key. In the PGP/GPG WoT, a user's identity is typically the user's name and her email address. Public keys of a user and certificates that vouch for the correctness of the key to identity mapping are published on a network of *key servers*, which are open to the public. Once published, neither

keys nor certificates can be deleted from the servers but only revoked (marked as invalid, compromised, etc.).

Typically, certificates are issued between persons that belong to the same social environment, i.e., among users that have already established personal contact. This environment is called a user's *community*. Certificates are often created when two users meet. This allows the users to verify their identities and public keys before they certify each other. Alternatively, certificates are often created at organized key signing parties at conferences, in universities, etc. Here the so called *introducer*, the user that signs the certificate, might need a passport or national identification card to verify the public key owner's identity before signing. The reason for this explicit identification is that the introducer does not know this person. In this case, the introducer acts as a kind of notary or certificate authority in PGP/GPG. Hence, the OpenPGP trust model can be seen as a *"super set of the centralized trust model we most often see in the X.509 world."* [31]

When one user wants to establish trust into a public key of another user, see Figure 3.3, no chain of trust starting at some central root CA is followed. Instead PGP/GPG tries to find a certificate path from the own key to the other key. PGP/GPG rates the strength of the found path using a *trust metric* customizable by the user. If the calculated trust value exceeds some threshold, the public key is considered trustworthy. For this process two types of trust are differentiated:

Introducer trust expresses the quality of the introducer. Every user estimates for herself how competent and careful an other user will act as introducer, i.e. how careful this user will verify the identity of another public key owner before she issues a certificate. The introducer trust is not public but stored locally by each user.

Public key trust is a value an introducer assigns to and stores in a certificate for a specific public key. The value expresses the verification quality of the public key, i.e. how certain the introducer is to have identified the public key owner correctly. This value is high, for instance, if the introducer has checked the identity of the key owner using an official document.

Various configurations of trust metrics are possible. A very careful user might chose to only trust public keys certified by users she directly knows and who have ultimate introducer trust. Less careful users might also accept a key that was certified by less trustworthy introducers or even any key that could be downloaded from a key server.

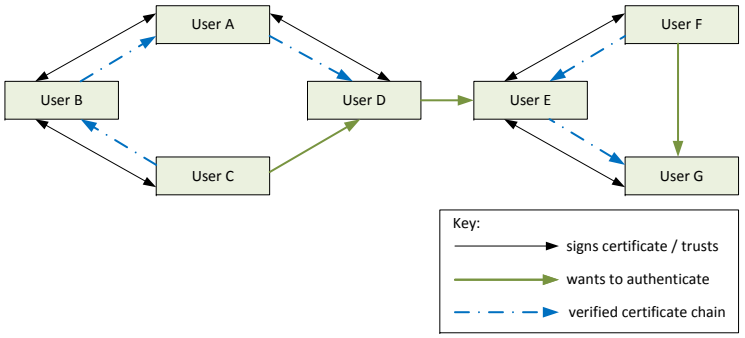


Figure 3.3: The PGP/GPG Web-of-Trust: User C (F) is able to establish trust in User D's (G's) public key as a chain of trust can be established between the users. As no chain of trust exists between User D and E, no trust can be established between them.

Benefits

The greatest advantage of the PGP/GPG trust model is that the users control the system. For this reason, a careful and experienced user can achieve a very high trustworthiness of the public keys used when a sufficient amount of suitable certificates can be evaluated.

Furthermore, this trust model is focused on the community of a user. This reduces the chance to accidentally trust the public key of users with a similar or ambiguous identity ("john.smith@xmail.com" vs. "john.smith@ymail.com"). The reason for this is that it is unlikely that two users with a similar name exist within the same community.

Drawbacks

Nevertheless, the high degree of involvement required by PGP/GPG is very demanding for the user. Therefore the PGP/GPG WoT is often regarded as a solution for experts and not for the average user. An inexperienced user that does not understand the concept of the WoT well enough might be tricked into trusting the wrong public key, which might lead to several problems.

Another major disadvantage of the PGP/GPG WoT are severe privacy concerns. The cause for these concerns is the public availability of user names, public keys and certificates. This is why everybody is able to retrieve keys and certificates of a specific PGP/GPG user. Besides containing the email address of the user, which might be abused to send Spam email, the harvested information can be used to determine the user's relationship to other users of the WoT. The found social relationships can be analyzed further in order to find communities, interest groups, the user's employer, etc.

A particularly problematic design decision of OpenPGP is that keys and certificates cannot be deleted from key servers but only revoked. So, the retrieved information about a user is no snapshot of the user's current status but a complete history of a user's social graph from the point in time she started to use PGP/GPG to present time.

Finally, as certificates contain time stamps, it is even possible (to some degree) to determine where a user was at a certain point in time when several information sources are correlated. For instance, it is easy to verify that a user has joined a specific key signing party at a conference.

3.3.4 Synthesis

Both PKI approaches have their strengths and their weaknesses. The X.509 PKI is quite easy to use, as it does not require any involvement of the user. Given that the participating PKIs are trustworthy, this trust model would be ideal for every user. However, examples exist that show that this assumption is not always true. The PGP/GPG WoT is highly flexible and offers high security provided that the users are experienced and able to operate the system well. For inexperienced users the complexity of the WoT might be too demanding and therefore even dangerous.

3.4 Access Control in Enterprise Networks

This section introduces several access control systems that are quite common in the enterprise environment.

3.4.1 Kerberos

Kerberos is described as an access control protocol that offers authentication and authorization in a networked infrastructure. A characteristic of the protocol is that authentication and authorization of a workstation/user that desires to access a service are two distinct operations.

First, the workstation requests a *Ticket Granting Ticket (TGT)* from the central *Authentication Server (AS)*. The TGT can only be used by the workstation if its user is able to provide her correct Kerberos password, which is needed to decrypt the TGT. In the second step, the workstation requests access to a service from the *Ticket Granting Server (TGS)*. The TGS checks if the user identified by the TGT is authorized to access the service and finally issues a *Ticket*. The Ticket then is used by the workstation to access the service.

If the user desires to access a different service, the whole protocol does not need to be performed again but the TGT can be reused to request another Ticket from the TGS. For this reason Kerberos can be seen as a single sign on solution.

Kerberos can also be used to perform authentication across so called *Authentication Realms*. Authentication Realms might for instance be assigned to branches of a company situated at different locations. A user does only need to be registered in one Realm. Kerberos of the local Realm takes care for the user's authentication and issues a Ticket that can be used in the other Realm to request access to a service from this Realm's TGS.

3.4.2 Remote Authentication Dial-In User Service (RADIUS)

The *Remote Authentication Dial-In User Service (RADIUS)* is a so called AAA service, i.e., provides centralized authentication, authorization and accounting. RADIUS is most often used to manage network access, e.g., to WLANs, dial-up networks, VPNs, etc.

RADIUS is based on the client/server paradigm. A service accessed by a device/user is called *RADIUS client*. The RADIUS client might for instance be the access point of a WLAN. The RADIUS client does not authenticate or authorize the device/user itself, but leaves the decision to the central *RADIUS server*. The authentication of the device/user is typically based on a username/password combination (knowledge). But it is also possible to leverage certified asymmetric keys for authentication. RADIUS also offers the ability to authenticate a device/user that belongs to a different Realm, which is for instance used in the eduroam initiative[2].

Besides managing network access, RADIUS can also be used to control access to services, such as the Apache web server. The web server acts as Radius client and leaves the authentication and authorization of the accessing user to the RADIUS server.

3.4.3 The eXtensible Access Control Markup Language

The *eXtensible Access Control Markup Language (XACML)* is a XML-based language for describing authorization and privacy policies. Version 2 of the standard was ratified by the OASIS Consortium in 2005 [32]. In 2010, version 3 of the standard [33] introduced several additions, which are not relevant in this thesis. Part of the standards is not only the language itself, but also the protocols and algorithms used to evaluate such policies and the XACML architecture, i.e., the XACML standards describe an entire authorization framework. Authentication of entities before authorization is not part of the XACML standard.

[2] https://www.eduroam.org

Architecture

The most important entities in the XACML architecture, see Figure 3.4, can be grouped into client applications, services and a central *Policy Decision Point (PDP)*. A service provides a resource requested by a client application, but also enforces (permits or denies) the access to it. For this reason the service has the role of a *Policy Enforcement Point (PEP)* in this architecture. The decision whether the client application is allowed to access a service is made by the central PDP.

Besides these main entities the *Policy Administration Point (PAP)* exists, which is of high interest in this thesis. The PAP is a central point that enables the network administrator to modify the *Policy Set*, which contains the access control settings of the network.

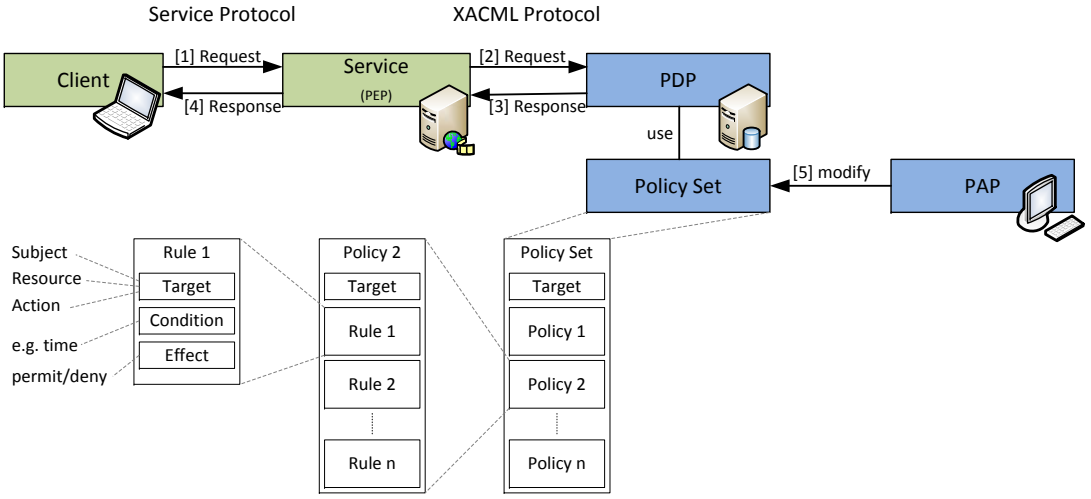


Figure 3.4: Overview: XACML architecture and Policy structure.

Protocol Flow

Upon a service request of a client application, see Step 1 in Figure 3.4, the service creates a *XACML request* that is sent to the PDP. A XACML request contains the identity of the *Subject* accessing the service (e.g., a user name) the identity of the requested *Resource* (e.g., the identity of the accessed service or even the identity of a certain file offered by this service) and the desired *Action* (e.g., read, delete, modify).

The PDP evaluates the request using the predefined Policy Set and additional meta-information (e.g. time) and sends its decision back to the service (Step 3). The XACML response contains the PDP's decision (permit or deny), which is finally enforced by the PEP (Step 4) by granting access or not.

Policy Sets

A XACML Policy Set consists of at least one *Policy*, see Figure 3.4. Each policy again contains at least one *Rule*. Policies, Rules and also XACML requests have a *Target* each, which is the XACML term for the combination of Subject, Resource and Action. Targets are used to match a XACML request to an applicable Policy in a first step and to an applicable Rule in a second step.

An extract of an example Policy Set is shown in Listing 3.1. Please note that several omissions were made to shorten the quite verbose XACML syntax. The example Policy Set contains several Policies, one is shown in the Listing, which control access to an individual service each. Each Policy again contains several Rules, one shown in the Listing, that

permits access to a specific user. Typically, Targets of Policies are quite generic and the Targets of Rules are more specific.

Please note: for the Targets of Policies and Rules it is possible to specify further attributes as shown in the example. For instance, if access to a specific action must be controlled, it would be possible to replace the wildcard operator `<AnyAction/>` (see Line 35) with a specific Action name, e.g. read, write, modify, etc.

```
1  <PolicySet [...]>
2  <Description>Example Policy Set</Description>
3  <Target/>
4    [...]
5    <Policy PolicyId="5" [...]>
6      <Target>
7        <Description>Policy for Service 5</Description>
8        <Resources>
9          <Resource>
10           <ResourceMatch MatchId="[...]:string-equal">
11             <AttributeValue DataType="[...]#string">Service5</AttributeValue>
12             [...]
13           </ResourceMatch>
14         </Resource>
15       </Resources>
16       <Subjects>
17         <AnySubject/>
18       </Subjects>
19       <Actions>
20         <AnyAction/>
21       </Actions>
22     </Target>
23     [...]
24     <Rule Effect="Permit" RuleId="17">
25     <Description>Rule for user Dave</Description>
26       <Target>
27         <Subjects>
28           <Subject>
29             <SubjectMatch MatchId="[...]:string-equal">
30               <AttributeValue DataType="[...]#string">Dave</AttributeValue>
31             </SubjectMatch>
32           </Subject>
33         </Subjects>
34         <Actions>
35           <AnyAction/>
36         <Actions/>
37       </Target>
38     </Rule>
39     [...]
40     <Rule Effect="Deny" RuleId="default"/>
41   </Policy>
42   [...]
43 </PolicySet>
```

Listing 3.1: Example: XACML Policy Set

Evaluation of a XACML Request

After receiving an example XACML request, see Listing 3.2, the PDP starts with policy evaluation. The PDP first determines which Policy should be applied. This is done by matching the Target named in the XACML request to the Targets of each Policy in the Policy Set. In this example, the Policy with ID 5 will be applied, as the XACML request contains `Service5` as Resource (see Line 11 of Listing 3.1).

After determining which Policy to use, the PDP will evaluate which rule is applicable. Again, the Target of the XACML request will be matched to the Target of each Rule. In this example the rule with ID 17 will be applied, as the Subject in the XACML request is `Dave` (see Line 30). The effect of the Rule is "permit" (see Line 24), which will result in the authorization of the client application.

```
1  <Request>
2   <Subject SubjectCategory="[...]:access-subject">
3    <Attribute AttributeId="[...]:subject-id" DataType="[...]:string">
4     <AttributeValue>Dave</AttributeValue>
5    </Attribute>
6   </Subject>
7   <Resource>
8    <Attribute AttributeId="[...]:resource-id" DataType="[...]#string">
9     <AttributeValue>Service5</AttributeValue>
10   </Attribute>
11  </Resource>
12  <Action>
13   <Attribute AttributeId="[...]#string">
14    <AttributeValue>read</AttributeValue>
15   </Attribute>
16  </Action>
17 </Request>
```

Listing 3.2: Example: XACML Query

Policy Administration

In contrast to the XACML syntax or the functionality of the PDP, the functionality of the PAP is not specified in the standards. In the simplest case a PAP is the text editor of the network administrator used to modify the Policy Set, see Figure 3.4, Step 5. A text editor is the most flexible but also the most difficult and error prone option a system administrator has to create or modify the Policy Set of her network. This is because the administrator must take care for the correctness of syntax and semantics of the created Policy Set herself.

Specialized XACML editors, such as the UMU-XACML-Editor [34], disburden the administrator in different aspects. The tool displays the XACML Policy Set in a clearly arranged, tree-like structure. For this reason the navigation through the Policy Set becomes user-friendlier. Additionally, the editor facilitates the Policy creation as the administrator is supported with XACML syntax suggestions and partially auto-generated XACML code. Finally, the editor takes care that a valid XACML Policy Set is created.

However, even specialized XACML editors are very technical and do not abstract over the XACML structure or assist the administrator with the semantics of the access control settings she needs to implement in XACML. For this reason XACML editors are quite comparable to an integrated development environment (IDE) for a programming language.

XACML Implementations

Basically two implementations for a XACML PDP corresponding to the XACML 2.0 standard exist. The first implementation was provided by Sun Microsystems [35]. The software was developed for a project starting in 2003, but seems to be orphaned these days. A second well maintained implementation is provided by the Heras-AF project from the University of Applied Sciences in Rapperswil. The Heras-AF project aims to *"provide, sustain and extend the de facto reference XACML 2.0 implementation named HERAS-AF XACML"* [36].

Full implementations of the XACML 3.0 standard are not yet available.

4. Identification and Authentication in Home Networks

The most fundamental requirement of the envisaged access control system for unmanaged networks is a scheme that identifies and authenticates entities that belong to these networks. This requirement has already been defined as Requirement RA "Secure Identification and Authentication Scheme" in Section 2.4.

In a (unmanaged) network domain, especially in a home network, several other requirements on the access control system exist that also influence the design of the needed identification/authentication scheme. These requirements were defined as Requirements RC "Decentralized Management" and RD "Access Control across Domains".

For this reason, an identification/authentication scheme is developed in this chapter that can be managed in a decentralized fashion and allows the authentication of entities across home networks.

Chapter Structure

The rest of this chapter is structured as follows. In Section 4.1 we define precise requirements for an identification and authentication scheme suitable for unmanaged network domains. The next Section 4.2 compares existing trust models and their implementations to these requirements. Our approach, to make the local network to the center of reference of the identification and authentication scheme is proposed in Section 4.3. Finally, Section 4.4 discusses and evaluates the findings of this chapter. The chapter is concluded in Section 4.5.

Acknowledgments

This chapter contains results developed by Blaž Primz[1] [37] and Simon Mittelberger [38] in their Diploma respectively Master's Thesis. These works were assigned and supervised by the Author in the context of his doctorate.

[1] Associated to University of Ljubljana. Performed his Diploma Thesis at Technische Universität München.

4.1 Requirements

Based on the analysis of properties of unmanaged networks at the example of the home environment in Section 2.2 we define requirements on the identification and authentication scheme for our access control system targeted to unmanaged networks.

- **RA.1: Hierarchical Identities:** The identity of an entity (device/service) must reflect the membership of this entity to the Domain the entity belongs to, e.g. a User Domain or a Home Domain.

  As a *belongs to* relationship exists between Domains of the *same* home network, i.e. a User Domain belongs to a Home Domain, the identity of an entity must also reflect this relationship.

  Finally the identity must also reflect the individual entity itself.

- **RA.2: Provable Identities:** Entities must be able to claim their identity and prove their membership in a specific Domain by authenticating to

  - entities that belong to the *same* Home or User Domain and
  - entities that belong to a *different* Home or User Domain.

- **RA.3: Autonomy and User-Centrism of Domains:**

  The identities of entities owned by a resident need to be controlled by this person, i.e.,

  - the owner of a Home Domain (home owner) manages identities of public entities and
  - the owner of a User Domain (resident of a home) manages the identities of her private entities.

- **RA.4: Scalability:** Due to the growing numbers of entities present in home networks, the identifier and authentication scheme must scale well.

- **RA.5: Offline Authentication:** Entities must be able to authenticate to other entities without the help of a third service that has to be online all the time.

- **RA.6: Versatility:** The identifier and authentication scheme must be flexible and must be applicable to human users, devices and services. Furthermore, the identifier and authentication scheme must be usable in different environments, e.g., in companies, universities, etc.

- **RA.7: Low cost:** The authentication scheme may not cause (high) costs. Especially recurring costs must be avoided.

4.2 Discussion of the State of the Art

In this section we discuss the suitability of authentication factors and existing Trust Models for the envisaged identification/authentication scheme.

4.2.1 Authentication Factors

A crucial design decision for our access control system is the selection of an authentication factor for claiming an identity. As we discussed in Section 3.2, three types of authentication factors exist.

Knowledge-based authentication obviously can satisfy many of the above listed requirements, namely Requirement RA.2 (provability), RA.3 (autonomy) and RA.6 (versatility), but has major drawbacks concerning usability and security. Think for instance of weak passwords selected by a user, see Section 2.3.1. Thus, knowledge-based authentication should be discontinued in future access control systems in the opinion of the Author.

Biometry-based authentication is too inflexible as it is only applicable to human users in quite special use cases (RA.6). Furthermore this technology is too complicated and too expensive, as special hardware is needed (RA.7). Thus, biometry-based authentication is not well suited as basis technology for access control, but might be an interesting complementary technology in certain suitable use cases.

The properties of the remaining possession-based authentication scheme depend in large part on the used key type, which is either a symmetric or asymmetric key. Authentication solutions based on symmetric keys are known to be difficult to handle, think about key exchange. For this reason we are convinced that the authentication scheme should be based on asymmetric keys. In this case, the properties of the identification/authentication scheme heavily depend on the used trust model.

Before we discuss the suitability of both trust models, it is worth to mention that possession-based authentication schemes can be implemented in pure hardware or software or as a "hybrid" system. For higher flexibility and lower costs, software-based authenticators, meaning asymmetric keys stored on a computer and used in software, can be used. If enhanced security is needed, specific hardware devices that contain and protect the authenticator can be used. Examples include cryptographic token, see Section 9.2.1, or a Trusted Platform Module, see Section 9.3.2.

4.2.2 Trust Models

After having identified asymmetric cryptography as most suitable base technology for our identification/authentication scheme, a suitable trust model needs to be found or created. As we have already explained in Section 3.3 two trust models are widely deployed today. In the following we discuss their suitability for our purposes.

4.2.2.1 X.509 Public Key Infrastructure

For this discussion we differentiate between private and public CAs/PKIs.

Public Certificate Authority/PKI

Entities that reside in a home network might obtain a certificate issued by one or several CAs that belong to a public X.509 PKI (see Section 3.3.2) located in the Internet, see Figure 4.1.

As a result all certified entities, either belonging to the same or different Home or User Domains, are able to automatically authenticate each other. This is because a certificate path originating from a trusted public (root) CA to the presented certificate exists (RA.2). This verification also works offline, as the public keys of the involved CAs are sufficient to verify a certificate chain (RA.5). Finally, the authentication scheme is versatile (RA.6) and also highly scalable when several different CAs are used (RA.4).

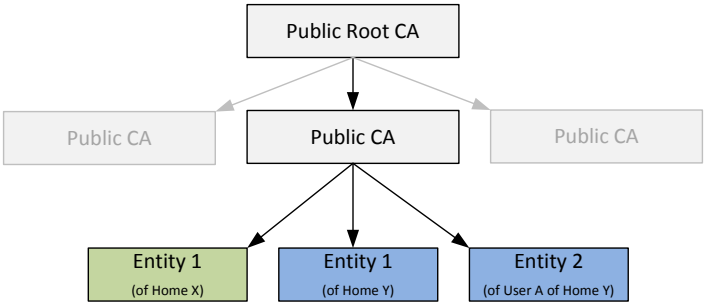


Figure 4.1: Public Certificate Authority/PKI

Unfortunately various problems exist. First, the use of a public CA directly conflicts with Requirement RA.3 (autonomy) as the management of identities depends on the public CA, which is clearly not in control of the Domain Owner. As a result, the users cannot estimate the trustworthiness of such a CA. Furthermore, even privacy concerns might arise, as the public CA would know each certified entity in every network. Additionally, costs must be expected when a certificate is created, renewed or revoked by the public CA, which contradicts RA.7. The approach also does not reflect the hierarchical structure of Domains (e.g. of Home/User Domains) and is not able to create a mapping between an entity and the Domain it belongs to. Finally, the identities certified by a CA are meaningless as the identity certified is only usable/known within its own local scope, i.e., its own home network (RA.1).

By these reasons, a public X.509 PKI is not suitable for our purposes.

Public/Private Certificate Authority/PKI

Introducing CAs "below" the above-described public PKI can solve some of the identified problems of X.509 PKIs in this context. These private CAs can be assigned to a Home or User Domain, see Figure 4.2, and be seen as the Home or User Domain's Identity. Home/User Domain CAs continue the chain of trust rooted in a (root) CA within the public X.509 PKI to entities that belong to a Home/User Domain.

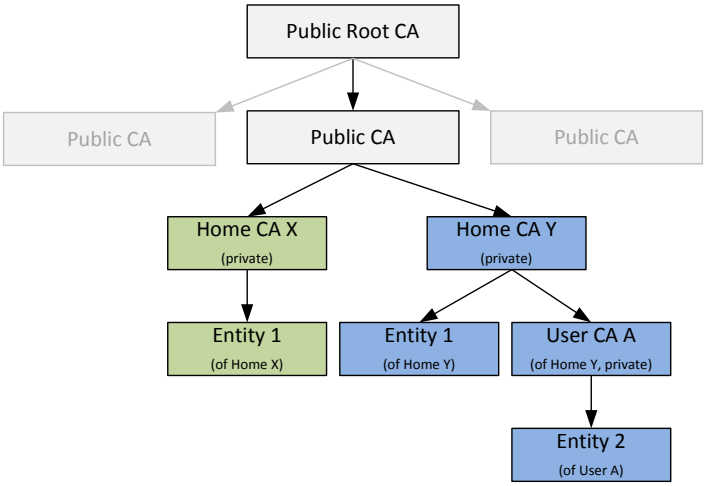


Figure 4.2: Public/Private Certificate Authority/PKI

As before, all certified entities are able to establish trust in public keys of other entities automatically. This is because a common (root) CA is present that acts as a root for the chain of trust between all entities (RA.2). Furthermore, the modified approach still fulfills Requirements RA.5 (offline authentication), RA.4 (scalability) and RA.5 (versatility).

Finally, the approach reflects the hierarchical structure of Domains and is able to map an entity to the Domain this entity belongs to (RA.1).

Autonomy (RA.3) is largely fulfilled as the Domain CAs are in full control of the Domain Owner. However, Domains still depend on public CAs to authenticate certificates issued outside of the own home. For this reason, the last requirement is not met as public CAs are expected to cause costs (RA.7).

4.2.2.2 PGP/GPG Web-of-Trust

An alternative way to establish trust into a public key of an entity is the PGP/GPG Web-of-Trust (WoT), see Section 3.3.3. The PGP/GPG trust model generally offers provable identities (RA.2), offline authentication (RA.5), is scalable (RA.4), versatile (RA.6) and can be used free of costs (RA.7).

As discussed before, the PGP/GPG WoT can be used in a way that mimics the role of CAs known from the X.509 PKI. In our context, a WoT-introducer might be assigned to a Home or User Domain, which would only certify the entities that belong to its own Domain. For this reason the autonomy of the Domain is ensured (RA.3). A different WoT-introducer assigned to each Domain will also establish the hierarchy layers that reflect the structure of home networks and enable the mapping between entities and the Domain it belongs to. Therefore a WoT-based approach also meets Requirement RA.1.

But, as we have discussed in the previous section, PGP/GPG has problematic characteristics concerning privacy. Being able to enumerate social relationships between Domains might facilitate targeted attacks on specific homes or cause other problems. Additionally, users might simply refuse this technology, as it does not protect their privacy.

Furthermore, today most software that performs authentication leveraging public key cryptography is based on X.509 certificates. Mature and stable implementations exist for instance, for SSL/TLS-based authentication between web servers and web browsers, for authentication to WLANs using EAP-TLS or for certificate-based VPN authentication, e.g. OpenVPN or IPSec. In contrast, available authentication solutions built upon PGP/GPG are mostly limited to email security.

4.2.2.3 Synthesis

We finally come to the conclusion that neither X.509 nor PGP/GPG satisfy the requirements we have defined on the identification and authentication scheme for networks domains. Nevertheless, the usage of a local X.509 PKI in a home with individual CAs for Home and User Domains controlled by the respective Domain Owner seems to be beneficial. But instead of using a central CA that establishes trust between different Domain CAs, a Web-of-Trust-like approach should be used. Trust between Domains can be established based on the idea to exchange certificates of “friendly” Domains. This hybrid approach is displayed in Figure 4.3.

4.3 The Home as Center of Reference

In the present section we explain the previously outlined idea to introduce a X.509 PKI in the home network in detail and derive an identifier scheme.

4.3.1 Identification and Authentication of Homes and Public Entities

Authentication

The first step towards our identification and authentication scheme is to introduce a X.509 CA that resides inside the home network. The so called *Home Domain CA* or simply *Home CA* is controlled by the home owner and issues public key certificates to public entities (devices and services) used by all residents of the home. A Certificate signed by a Home CA

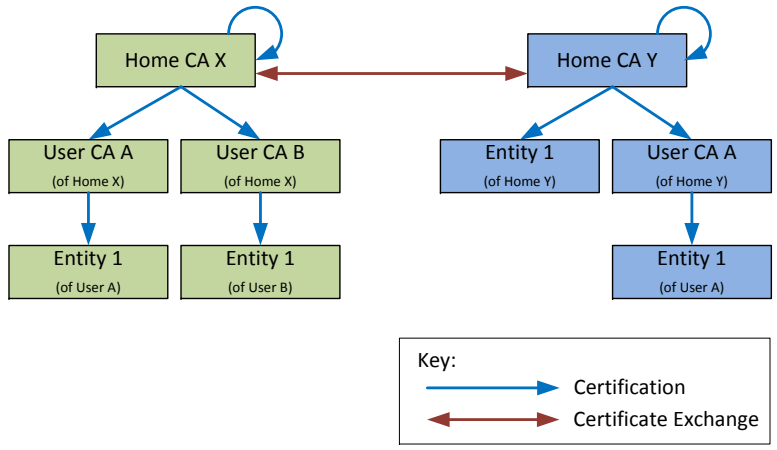


Figure 4.3: Hybrid Trust Model: Within a home network a local X.509 PKI is deployed, which is the basis for the local identification and authentication scheme. Authentication of entities that belong to different home networks can be enabled using a WoT-like approach, i.e. by exchanging certificates between homes.

1. asserts that a public entity is a valid member of this home and
2. binds the entity's human readable identifier to its public key.

Identification

A public key could naturally be used as identifier of an entity. In our scenario the public key of the Home CA could act as an identifier for the home network and the public key of an entity might act as identifier for the entity.

A drawback of using the public key directly as identifier is that such an identifier is not hierarchical and too long. Typically public keys have lengths of several hundreds or thousands of bits. This might be inconvenient to handle or have other drawbacks, for instance, a public key used as identifier could bloat database tables or network messages. For better handling, we therefore propose to shorten the public key using a cryptographic hash function, such as SHA-1 or SHA-512.

According to this explanation, the identifier of the home network, the so-called *Home ID*, is computed as follows:

$$HomeID \;=\; H(PubKey_{HomeCA})$$

The Home ID is a unique, *self-certifying* identifier. Self-certifying denotes the property of an identifier that it does not need to be accompanied by a certificate that maps this identifier to a public key. The reason for this property is obviously that the identifier was derived by hashing from a public key.

For public entities that belong to a home network we propose an identifier that consists of the Home ID plus a second identifier-part. This second identifier-part is computed analogously as the Home ID by simply applying a cryptographic hash function (H) to the public key of the public entity. Finally, both identifier-parts are concatenated (.) in order to obtain the needed hierarchical identifier.

According to this explanation, the *PublicEntityID* is computed as follows:

$$PublicEntityID \;=\; H(PubKey_{HomeCA}).H(PubKey_{PublicEntity})$$

The identifier scheme presented so far does not consider the users and their private entities. Only identities for the home and its public entities can be expressed so far. This limitation would have no drawbacks in homes where only one person lives. This is because the home can be equated to the person that owns it. In more complex setups, meaning homes inhabited by multiple persons, this is not possible anymore and the scheme presented so far is insufficient and must be extended. Therefore, we propose to extend the basic identifier scheme with a hierarchy layer for the users.

4.3.2 Identification and Authentication of Users and Private Entities

Authentication

Next to the Home CA we propose to introduce individual *User CAs* for every resident of the home. User CAs are certificate authorities assigned to and controlled by an individual user. The Home CA must certify a User CA like a public entity. This means the User CA obtains a valid public key certificate that

1. asserts that the user represented by a specific User CA is a valid member of this home,
2. binds the user's human readable identity to her public key and
3. makes the User CA to a valid sub CA within the home.

As User CAs are valid sub CAs within the home, they are allowed to issue public key certificates to private entities that belong to the Domain owned and controlled by the resident. A certificate signed by a User CA

1. asserts that a private entity is a valid member of this User Domain and
2. binds the entity's human readable identity to its public key.

Identification

A User CA is the basis for identification in a User Domain. As a User CA it is always assigned to a single person it is now possible to create identifiers for the users as well. The *User ID* is generated analogously to a PublicEntityID using a cryptographic hash function and concatenation:

$$UserID \;=\; H(PubKey_{HomeCA}).H(PubKey_{UserCA})$$

Finally, a new identifier type for entities that belong to a certain user/User Domain can be created, the *PrivateEntityID*:

$$PrivateEntityID \;=\; H(PubKey_{HomeCA}).H(PubKey_{UserCA}).H(PubKey_{Entity})$$

4.3.3 Example

A simple example of identities in a home network is shown in Figure 4.4. The Home Domain is represented by Home CA Y whose public key is the basis for the Home ID (`aab..732`). One public entity is registered directly to the Home Domain. Its public key and the Home ID are the basis for this entities ID (`aab..732.36a..96e`). Additionally, one resident runs her private User Domain, which is represented by User CA A. The ID of this Domain consists of the Home ID concatenated to the hashed public key of the User CA A (`aab..732.773..55d`). Finally a private entity exists, which is registered to User Domain A. The entity obtains its ID from the User ID concatenated to the hashed public key of the entity (`aab..732.773..55d.9d3..112`).

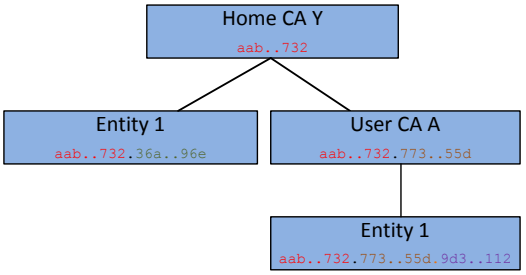


Figure 4.4: Example: Hierarchic Identities within a home network.

4.4 Discussion and Evaluation

In the following we discuss and evaluate the contributions of this chapter.

4.4.1 Fulfillment of Requirements

The design of our identifier and authentication scheme presented in this chapter meets all requirements we defined in Section 4.1, which we want to explain in the following.

As specified in Requirement RA.1, the identifiers derived from the chain of public keys (Home CA - User CA - Entity) are hierarchic and express the membership of a private entity to a User Domain (resident), or the membership of a User Domain or public entity to a Home Domain (home).

The X.509 PKI (Home CA - User CA) we introduced in the home network is furthermore the center of reference for secure authentication (RA.2). A locally signed certificate can be used to claim an identifier among all entities that trust the CAs of the local PKI, i.e., the Home CA.

The presented identification/authentication scheme also fulfills the requested autonomy of Domains and is user-centered (RA.3). A resident controls her personal User Domain and her private entities. The home owner controls her Home Domain and her public entities.

The identification/authentication scheme is also highly scalable, as it is based on a X.509 PKI (RA.4).

As the Home CA acts as root of trust for the local PKI it is sufficient to possess the Home CA's self-signed certificate in order to validate the certificate chain presented by an entity. For this reason, no authority needs to be online all the time and authentication can also be performed offline (RA.5).

As our identifier and authentication scheme is based on asymmetric cryptography and X.509, the scheme is also highly versatile (RA.6). Asymmetric cryptography and X.509 is universally applicable, widely deployed today (TLS/SSL), can be used by human users and entities (devices, services) and finally can be implemented in software and hardware (cryptographic token, TPM, etc.) when extra security is needed.

Finally, as CAs are used that reside within the home, no costs for the certification, renewal or revocation of certificates are to expect (RA.7). The introduced CAs themselves can be implemented based on free and open source software, such as OpenSSL, and be hosted on inexpensive hardware comparable to today's home routers.

4.4.2 Further Benefits

The presented hierarchical, self-certifying identifiers enable addressing of entities within or between home networks. Home networks might form a Peer-to-Peer (P2P) network

and constitute a Distributed Hash Table (DHT). An entity that wants to access a service located in a remote home can use the hierarchical identifier of the service, more precisely, the Home ID is sufficient, and request the current IP address of the home network from the DHT, see Steps 1 - 4 in Figure 4.5. After this initial lookup, the service requester can directly send a request to the Home Network using the IP address just received, see Step 5. Within the home network a comparable P2P-based solution for address look up could be implemented or the Domain Name System (DNS) might be used. No matter which technology is used, the request can be forwarded to the requested service (Step 6) and the request will be answered, see Step 7. The just outlined system and its implementation is detailed in the work of Blaž Primz [37].

Our approach has various similarities to the *Host Identity Protocol (HIP)* [39]. HIP also uses identifier derived from public keys, so called *Host Identity Tag (HIT)*. A difference to our solution is the hierarchic structure of our identifier.

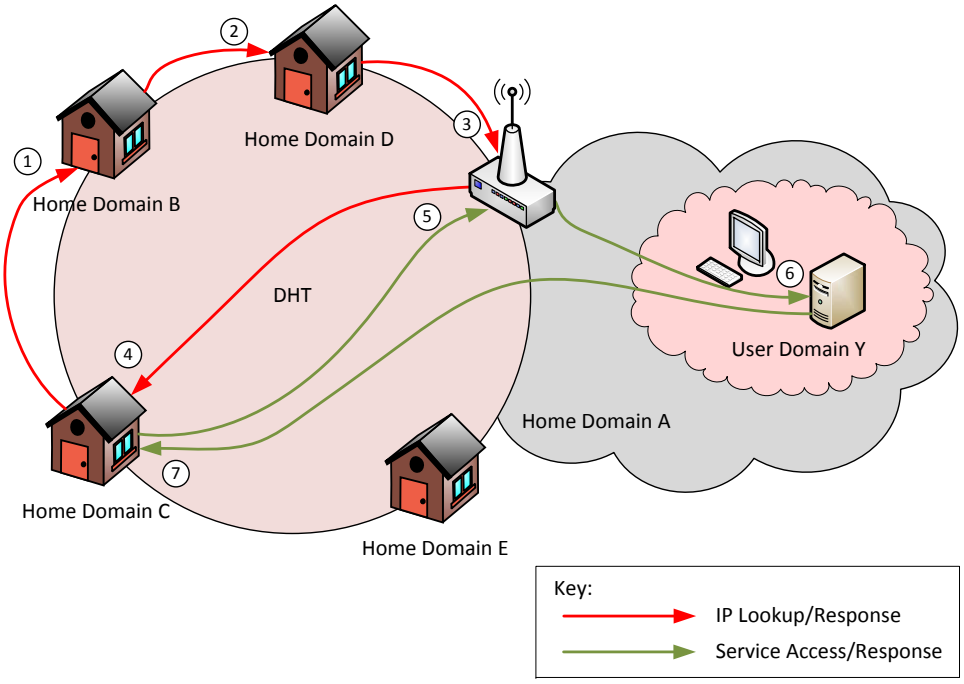


Figure 4.5: Addressing and Routing between Home Networks: Home Domains may join a DHT, which allows to lookup a home network's current public IP address. After the IP lookup, service requests can be sent directly to the home and forwarded to the desired service.

4.4.3 Applicability

The presented scheme can be applied to every networked environment or scenario and is not limited to home networks with only two hierarchy layers (Home Domain/User Domain). In a company the identifier scheme can be extended easily to express the company itself, a department, an individual employee and the entities managed by her by introducing a Domain CA in each hierarchy layer. For this reason, the identifier/authentication scheme is generic and can be adapted to other scenarios.

4.4.4 Open Issues

At this point we need to point out that RA.2, authentication across domains, is not completely satisfied yet. Currently only entities that belong to the *same* home are able to authenticate each other. This is no conceptual weakness or limitation, but an open

issue that will be addressed in Chapter 6 by a user-friendly approach to establish Trust Relationships between different homes.

Furthermore, introducing a PKI inside the home raises the question how the inexperienced users that often exist here can be enabled to manage the identities of their entities. For this purpose a solution is needed that enables inexperienced users to effortlessly generate and certify asymmetric keys as basis for identities for their entities. This issue is addressed in the following Chapter 5.

4.5 Conclusion

In this section we discussed how an identification and authentication scheme for unmanaged networks, such as home networks, can be created. The proposed solution is based on a private X.509 PKI that exists within the local network. With this step, the local network becomes to the center of reference of the local identification and authentication scheme. From the public key certificates issued by the local CAs, self-certifying and hierarchical identifiers can be derived using a cryptographic hash function and concatenation.

The found solution is superior to existing PKIs or trust models, such as X.509 or PGP/GPG, as the presented approach does not have the drawbacks, i.e. privacy and trust issues, of these technologies.

We also discussed that the proposed scheme is applicable to different network types other than home networks. We finally pointed out that the presented identities can be used easily for addressing and routing between different networks.

Key Findings and Contributions

▷	Several independent **Domains** must be differentiated in a home network in order to reflect its social and administrative structure. A **Home Domain** represents the home itself. Individual **User Domains** represent every user. Devices and services (entities) possessed by the home or by residents are members to the corresponding Domain.
▷	An identification and authentication scheme suitable for home networks must reflect this structure in order to allow user-centered management of access control.
▷	Public PKIs (X.509 PKI or PGP/GPG Web-of-Trust) are not suitable in this scenario due to trust and privacy issues.
C4.1	**Authentication Scheme** for Home Domains, User Domains and entities rooted in the local network: A **Home CA** represents a Home and acts as root of the local, private X.509 PKI; **User CAs** represent User Domains.
C4.2	**Hierarchic, self-certifying and unique identities** for homes, residents, and entities derived from the certificate or certificate chain of a Home CA, a User CA, or an entity certified by the Domain CA.
C4.3	**Hybrid Trust Model** combines benefits of X.509 (compatibility to existing software, ability to form hierarchical PKIs) and a Web-of-Trust-based Trust Model (user-centrism).

5. User-Friendly and Secure Identity Distribution

The identification and authentication scheme presented in the last chapter is based on the approach that every device, service and user possesses an asymmetric key pair certified either by the local Home CA (certifies public entities and the Domain CA's assigned to residents) or by a User CA (certifies private entities). The major problem when we want to introduce this scheme, especially to an unmanaged environment such as a home, is that average users cannot be expected to be able to generate asymmetric key pairs and to take care for the certification of a public key on their own. Furthermore, most users will not understand the theoretical concepts of certification and asymmetric cryptography, which might be an obstacle for the acceptance of this technology.

For this reason, all technical details and difficult to understand concepts need to be hidden from the user in order to achieve high user-friendliness, which has already been defined as Requirement RE in Section 2.4. Therefore, this chapter addresses the question how average users can be empowered to distribute asymmetric key pairs certified by their Domain CA to their entities. The answer to this question is a user-friendly and secure identity distribution service we present in the following.

Chapter Structure

In the following Section 5.1 we first introduce the basic idea to create a "guided", semi-automatic Registration Service and explain some architectural issues concerning this service. In Section 5.2 we define precise requirements on the highly security critical protocol that distributes identities to entities. The Registration Protocol is finally explained in Section 5.3. Section 5.4 discusses and evaluates the findings of this chapter. The Chapter is concluded in Section 5.5.

Acknowledgments

This chapter contains results developed by Sunil Kumar Ghai[1], Simon Stauber and Simon Mittelberger during their work as research assistants. These works were assigned and supervised by the Author in the context of his doctorate.

[1] Associated to the Delhi College of Engineering. Performed an internship at the Technische Universität München.

5.1 Approach and Architecture of the Registration Service

Registration Metaphor

As we have explained in the introduction of this chapter, many inexperienced users will not understand concepts like certification or asymmetric cryptography. For this reason we introduce the easy to understand *Registration* metaphor: Every entity in the home needs to be *registered* to one of the Domains that belong to the home in order to make it to a valid member of the home network. Public entities are registered to the Home Domain; a private entity is registered to its owner's User Domain.

Behind the scenes all complicated technical steps are performed automatically. This means that a new key pair is created for the entity, the appropriate Domain CA issues a certificate for this key, and finally the certificate is transported to the entity. Later, the certificate can be used to authenticate the entity towards the home network itself or to services hosted in the home. The certification happening in the background of the Registration process or the fact that a certified key is used for authentication are side effects the user does not need to care about or even understand.

The Registration metaphor is fairly easy to understand, as various other technologies users are already familiar with use a similar idea, which is to "pair" devices in order to connect them. Examples include the setup of Bluetooth connections between devices or *Wi-Fi Protected Setup (WPS)*.

Overview on the Registration Infrastructure

Figure 5.1 depicts an overview of the soft- and hardware components involved in the registration of a device to a Domain and a simplified version of the Registration Protocol. In Step 1 the device sends a Registration Request. The Domain Owner is notified about this incident in Step 2 and decides whether to permit or deny the request in Step 3. When permitted, the certificate signed by the Domain CA is sent to the device in Step 4.

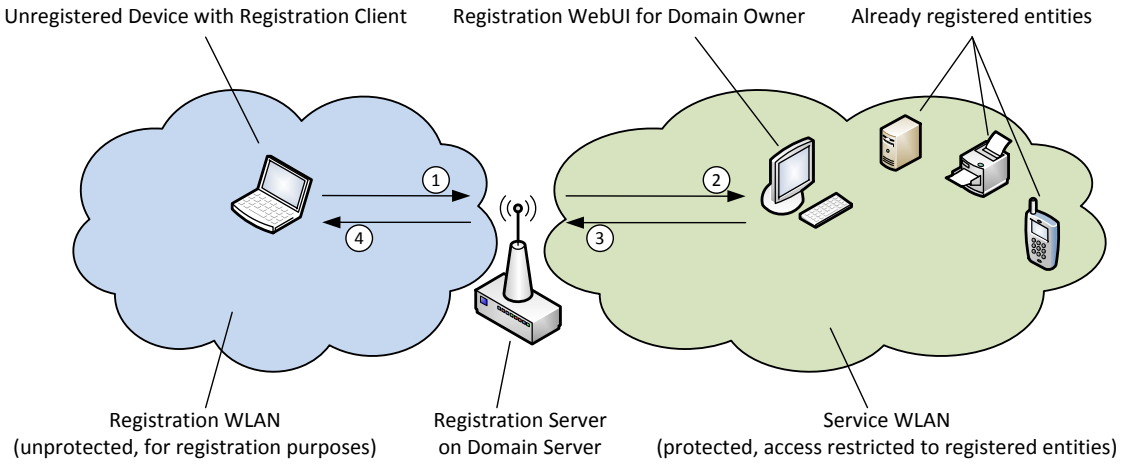


Figure 5.1: Simplified Registration Protocol of a device to a Domain.

The central hardware component required by our system is called *Domain Server*. The Domain Server can be thought of as next generation home router. Its detailed architecture, additional services, and protection mechanisms are explained later, see Chapter 10ff.

On the Domain Server the *Guided Security Management Systems* of all Domains that belong to the home are hosted. The Guided Security Management System consists of different service components that are designed and implemented in this thesis in order

to allow users to manage access control to their Domains securely and effortlessly. The service component discussed in this chapter is the *Registration Service*, a semi-automated front end or *registrar* for a Domain CA. Other service components of the Guided Security Management Systems will be introduced in subsequent chapters.

Registration is an important and security relevant process as valid certificates are distributed to new entities. Therefore, the owner of a Domain needs to be able to control the Registration Server of her Domain. For this purpose a graphical user interface (GUI), the *Registration GUI*, is provided. The Registration GUI is an easy to use web application that can be accessed using a web browser installed on a device that was already registered to this Domain, i.e., which is authorized to access the Registration GUI of this Domain.

On the side of devices the users are supported with a small application called *Registration Client*. New devices, e.g., laptops, can be equipped with the Registration Client simply by installing this component from an USB memory stick. For higher convenience, the Registration Client might also be distributed using a web server running on the Domain Server. This web server is accessible via the open Registration WLAN.

Registration and Service WLAN

The Registration Client installed on a device and the Registration Service need a communication medium in order to exchange information. Several technologies including Near Field Communication (NFC), Bluetooth and WLAN are well-suited candidates. In fact, NFC would provide the probably most intuitive and natural registration experience. The registration could be "magically" started when the device that should be registered is held closely to the "registration point" that represents a Domain CA. Unfortunately, apart from smart phones not many devices support NFC yet.

A better option is to use Bluetooth or WLAN. Both technologies have a long range, which is convenient for the users, and are supported by almost all computing devices today. However, WLAN seems to suit better to the idea to distribute the Registration Client using above mentioned web server. For this reason, we decided to use WLAN as default communication medium for Registration of devices. However, it is worth to mention that the Registration Protocol can be performed over Bluetooth or any other wired or wireless communication medium as well.

The next open question is how the needed WLAN can be secured in a way so that it can be accessed by devices that desire to perform Registration. No cryptographic secrets are exchanged between network and the device yet. This is quite problematic, as users should not be bothered to enter long and complicated passwords that are used to protect the WLAN.

For this reason we propose to deploy two WLANs with distinct tasks in the home. A *Registration WLAN*, which is exclusively used as communication medium for devices during Registration and a *Service WLAN*, which provides access to the "real" home network, i.e., to all services hosted in the home and also to the Internet.

The Registration WLAN is not protected using standard mechanisms, such as WPA2 or VPN. Instead, we decided to leave the Registration WLAN open (entirely unprotected) and secure the messages exchanged during registration on the application layer of the ISO/OSI model. For this purpose, a session key is generated first and then used by AES to encrypt exchanged messages. The advantage of this innovative approach is that the security of the Registration Protocol is agnostic of the underlying communication medium.

After a new device is successfully registered to a Domain that belongs to the home it can use its certified asymmetric key pair to authenticate itself to the home's Service WLAN.

For this purpose an access point is needed that uses WPA2 in enterprise mode, i.e., that authenticates entities using EAP-TLS leveraging their certificates. The RADIUS server, which performs the authentication of the accessing device on behalf of the Access Point, can be hosted on the Domain Server as well, see Figure 5.2. After accessing the network, the devices certificate can also be used to authenticate itself to other services offered in the network.

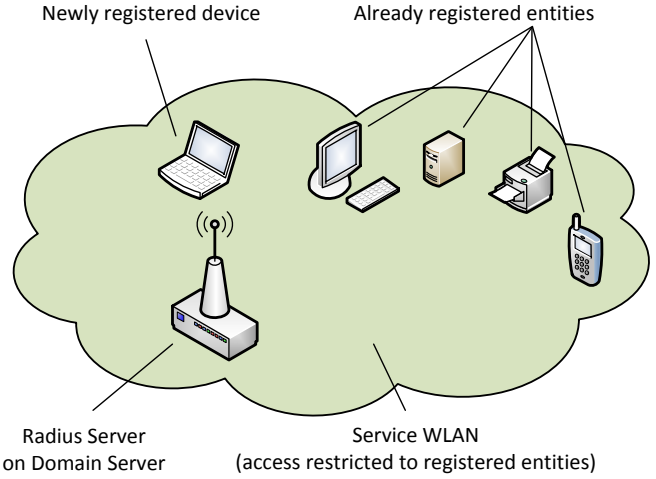


Figure 5.2: After Registration the new device may access the Service WLAN after EAP-TLS authentication using the just received certificate.

Please note: For registering new *services* in a Domain, the users can be supported in a similar fashion. For this purpose, the installer application of a service must be extended with functionalities similar to the Registration Client. The Registration of a service will then become part of the service install routine. Compared to a device, which first needs to be connected via the Registration WLAN to the Registration Server, the Registration Client for a service does not need to care about connectivity. This is because the service will be installed on a device, which is already connected to the home network.

5.2 Requirements

In this section we present requirements considering the security and user-friendliness of the protocol that implements Registration.

Security

The registration of an entity is a highly critical process, as a valid certificate is deployed to an entity that later asserts the membership of this entity to a Domain. Every precaution must be taken in order to guarantee that only a legitimate entity receives a valid certificate.

We therefore briefly discuss some possible attacks and define requirements on protocol security.

- **R1: Identification/Authentication of the User Registering an Entity:** An adversary might try to obtain a valid certificate for a device controlled by her from the Registration Service of a Domain. Therefore, Registration may be performed only after this user could identify/authenticate herself to the Domain Owner.
- **R2: Protection of the Registration Session:** An adversary might act as a man in the middle and inject her public key into a running Registration Session in order to trick the Registration Service to certify her key. Therefore, all messages transmitted between Registration Client and Service need to be encrypted/authenticated.

- **R3: Authentication of the Registration Server:** An adversary might try to trick an user to register her device to a "faked" Registration Service that claims to belong to the user's Domain. After the Registration with this service, the device might connect to a Service WLAN the adversary offers. As a result the security of the device or the privacy of its user might be endangered, as the attacker can eavesdrop the communication. For this reason it is also required to authenticate the Registration Server.

- **R4: Resistance to Denial-of-Service:** An adversary might flood the Registration Server by sending faked Registration Requests. As the Domain Owner must permit each Registration Request (R1) this attack might become cumbersome and impair the overall user experience and usability of the Registration Service. Therefore, the protocol must be designed in a way that Denial-of-Service like attacks are impossible.

User-Friendliness

The Registration Protocol has a significant influence on the user-friendliness of the Registration process. Therefore the protocol must fulfill the following requirements on user-friendliness:

- **R5: High Degree of Automation:** All complicated protocol steps that do not require input of human users must run automatically. Especially complicated configuration tasks or tasks that require technical skills must be hidden and transparent to the user.

- **R6: No Loss of Control:** Automation has often the drawback that users might feel like losing control. So it is important to give the user the feeling that she is in control. Therefore, the system needs to give feedback and require the input of the Domain Owner when important decisions need to be made, e.g., permitting a Registration.

- **R7: Low Interaction Required:** The amount of interaction between Registration Service and Domain Owner must be limited. Otherwise the system might be regarded as too noisy and will be ignored, which would conflict with the next requirement.

- **R8: High Responsiveness:** The Registration Protocol must run quickly. Long waiting times until the registration of an entity is completed must be avoided. As no processes are involved that require time consuming operations it is to expect that the run time of the Registration Protocol is mainly influenced by the reaction time of human participants. Therefore the protocol must avoid repeated interlocking/waiting situations between human users.

5.3 Registration Protocol

The Registration Protocol is depicted in Figure 5.3. For simplicity, the Registration Server and Registration GUI are depicted as only one element. Depending of the exact scenario, some protocol steps can be omitted or are done implicitly when human beings that participate in the protocol, namely the *Domain Owner* and the *User*, are the same person. This is the case when 1) the home owner herself registers a public entity to the Home Domain or 2) when a resident registers a private entity to her personal User Domain. In both cases the communication between Domain Owner and User (which means Steps 1 and 8) are obviously not needed. Additionally, Steps 10 - 14 can be omitted when a service is

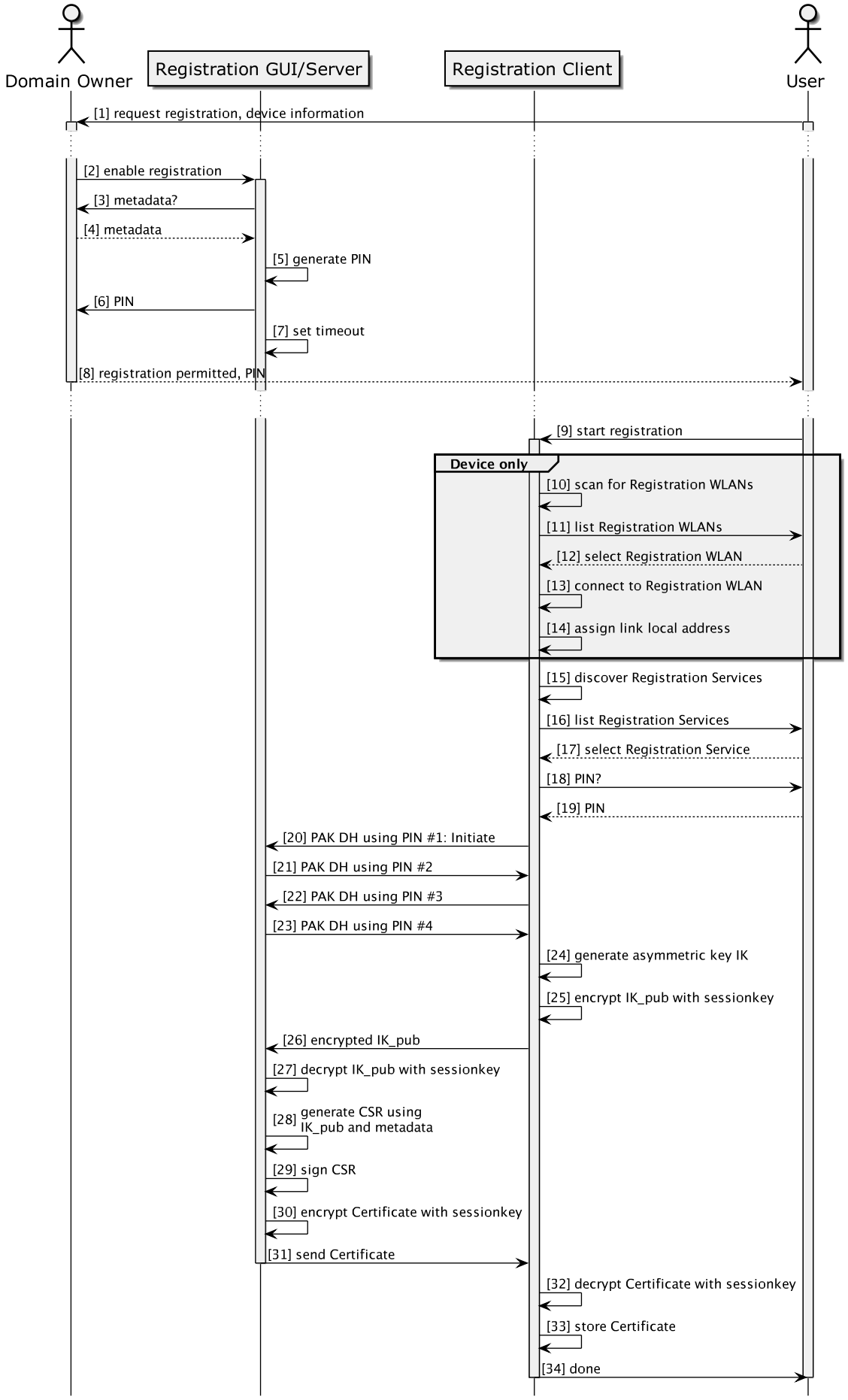


Figure 5.3: Registration Protocol

registered. This is, as discussed, because connectivity to the Registration Service does not need to be established anymore in this case.

In Step 1 the resident asks the Domain Owner (home owner) for permission and also tells the Domain Owner which kind of entity should be registered. This request either can be made personally, or via telephone, email, instant messenger, etc. However, this first step acts as an implicit identification/authentication of the user registering the new entity.

After deciding whether the user should be allowed to registered the entity or not, the Registration process is activated by the Domain Owner in Step 2, i.e., the Registration Server is started and a new Registration Session is created.

After enabling the Registration Server, the server requests the Domain Owner to provide meta-data about the new entity. Typically, a human readable description of the entity, e.g., "TV set located in living room" or "family PC", will be provided (see Steps 3 - 4). Additional information might be the name of a contact person, etc.

Now the Registration Server computes a short human memorable randomized PIN (Step 5), which is later used to claim this Registration Session, displays this PIN to the Domain Owner (Step 6), starts a timeout, and finally waits for an incoming Registration Request (Step 7). If the timeout expires, the Registration Session is closed and needs to be reactivated.

In Step 8 the Domain Owner communicates the PIN via a secure channel to the User. This can be done in person or via other mechanisms that ensure that only the User that requested the Registration will obtain the PIN, e.g., via phone, a text message, etc. After this step, the Domain Owner is not involved in the remaining Registration process any more.

After the User obtained the PIN, the actual Registration process starts. The user activates the Registration Client (Step 9) that starts to search for open Registration WLANs (Step 10) in the area. When finished, the results are displayed to the User (Step 11) who selects the appropriate Registration WLAN (Step 12). For this purpose, the Registration WLAN needs to be named reasonably, e.g., "regWlanSmithFamily". Finally the Registration Client connects the device to the selected Registration WLAN (Step 13).

After connectivity is established, the Registration Client performs IP auto configuration and configures the WLAN device with the received link local address (Step 14). Subsequently the Client discovers active Registration Servers (Step 15) using the mDNS/DNS-SD protocol suite [40] and displays the results to the user (Step 16). In most cases only one result will be presented, as the Registration Servers are offline for most of the time. Nevertheless, a reasonable name, such as "regServSmithFamily" should be used to name a Registration Server.

After the User selected the desired Registration Server (Step 17), the Registration Client requests the PIN the User has received before from the Domain Owner (Step 18). After entering the PIN (Step 19), the remaining process runs completely automatically without any further interaction with the users.

The Registration Client now uses the PIN as input for a *password-authenticated key agreement protocol (PAKE)* to derive a shared session key between Registration Server and Client. Various PAKE protocols exist. We have selected the *Password-Authenticated Key Diffie-Hellman Exchange (PAK)* protocol, which is described in RFC 5683 [41]. PAK provides *"mutual authentication, based on a human-memorizable password, to the basic, unauthenticated Diffie-Hellman key exchange"*. This session key is derived in Steps 20 - 23 and is later used to protect (encrypt/authenticate) the messages exchanged between Registration Client and Server.

In Step 24 the Registration Client generates the asymmetric key pair of the entity, which is called *Identity Key (IK)*. The IK is the basis for the cryptographic identity of this entity. Now the Registration Client encrypts the public part of the IK using the session key established before (Step 25) and finally sends the encrypted IK to the Registration Server (Step 26).

The Registration Server decrypts the incoming message using the negotiated session key (Step 27) and creates a certificate signing request based on the received public part of the IK and the meta-data entered by the Domain Owner before (Step 28). Then, the CA of the Domain will sign the CSR (Step 29). Finally, the resulting certificate and the self-signed certificate of the Home CA are encrypted (Step 30) and sent to the Registration Client (Step 31).

As last steps the Registration Client decrypts the received message (Step 32), stores the certificates locally (Step 33) and finally displays a success message to the user (Step 34). From now on the device is a valid member of the Domain and may authenticate, for instance, to the homes Service WLAN.

Please note: If an entity is registered to a User Domain, the entity receives besides its own certificate the Home and User CA certificates.

5.4 Discussion and Evaluation

5.4.1 Fulfillment of Requirements

The proposed Registration Protocol fulfills all requirements on security and user-friendliness we have defined in Section 5.2.

The Domain Owner identifies/authenticates the user that requests registration explicitly and authorizes (allows) her to perform the registration by handing out the PIN (R1). This PIN is later used by the PAK protocol to generate a secret session key that protects the Registration Session. This key agreement process will only succeed when both involved parties (Registration Client/Server) are able to present the identical PIN to the PAK protocol. So a mutual authentication between Registration Server and Client is achieved (R3).

The messages exchanged between Registration Client and Server that contain important information, especially the public part of the IK of the new entity (sent in Step 26), are encrypted/authenticated with the previously generated session key before sending. As already discussed, the session key can only be generated by the legitimately participating entities. So no adversary is able to interfere with the protocol and, for instance, replace the IK sent from a Registration Client installed on a legitimate device with another public key, which is possessed by her (R2).

The Registration Protocol was designed to be resistant against Denial of Service attacks. We can exclude this type of attack because the Domain Owner needs to activate registration manually. This is an interpersonal process between two human beings that already know each other. After the activation of the Registration Server, an adversary could notice the availability of the Registration Server and send illegitimate Registration Requests. As these requests cannot be authenticated from the Registration Server, as the adversary does not know the PIN, they are silently discarded (R4).

As required, all protocol steps that do not need the input from human users are completely automated (R5). Additionally no surplus messages that could annoy the users, especially the Domain Owner, are sent or displayed (R7). The protocol is also designed in a way that interlocking of processes that results from repeated waiting for the input of human

users is avoided. The only required interaction between human users and the system is related to permitting the Registration process (R6).

Finally, the Domain Owner has full control of the Registration Service, as she needs to activate the Registration Server and permit the Registration Request (R8). In this way we can guarantee that only intended entities are registered to the Domain.

5.4.2 Further Attacks

As discussed before, it can be excluded that an adversary is able to illegitimately register a device by exploiting a technical process. The only way that remains is that a dishonest resident of the home assists the adversary. The resident could ask the Domain Owner to permit the Registration of an entity and pass the PIN received from the Domain Owner to the unintended user who then might perform the Registration of her device. This is a social attack and cannot be ruled out technically.

Another variant of this attack would be that a dishonest user that has access to an already registered public entity extracts this entities Identity Key and certificate and passes both to an unintended user. This attack and other related attacks that involve the abuse of certified keying material are discussed and resolved in Chapter 13.

5.4.3 Applicability

The Registration Protocol is applicable to various other environments or scenarios and not limited to the home environment. The Registration Service might deploy certified asymmetric keys in any type of dynamic environments, e.g., in hotels, at trade fairs or conferences. Devices of customers can use these certified keys for WLAN access, etc. As we have described, this technology is secure and simple to use and will decrease administrative overhead strongly.

5.4.4 Open Issues

Up to now we did not explain how inexperienced users can be enabled to manage their Domain Server and perform maintenance tasks needed to create Domains or ensure the resilience of their Domain. These issues will be discussed and resolved in Chapter 11.

5.5 Conclusion

Introducing a PKI inside a home network is the first step towards secure identification and authentication. The distribution of identities, meaning asymmetric keys certified by the local PKI, to entities that belong to this network is the second step. Especially in those networks where inexperienced users are present and no help is available, the second step will become a major obstacle. Without help, the identifier/authentication scheme cannot be used, as users alone are not able to create and distribute identities. Additionally, users might make serious mistakes when they distribute identities to wrong entities, which would decrease the security of the access control system.

For this reason we introduced the easy to comprehend Registration metaphor and designed and implemented a user-friendly and secure Registration Service. This Registration Service provides all needed functionality to distribute identities to entities that belong to the home, to secure this process, and finally to guide the users through all difficult to comprehend or perform steps.

After the distribution of locally certified asymmetric keys to entities, state of the art and highly secure enterprise grade authentication mechanisms can be used for the first time in homes or other unmanaged environments, like small companies.

The state of the art of home networking does not offer a concept comparable to ours. The currently existing system that has most similarities to the mechanisms presented in this chapter is WPS. Here, a secret key needed for WLAN authentication is exchanged semi-automatically between WLAN Access Point and a device. In contrast to WPS, our approach distributes identities that are generic and can be used for many other purposes than for only WLAN access control.

For this reason, the overall benefit gained by the introduction of our identification/authentication scheme and the Registration Service can be considered as high. Before, users had to take care for the passwords used for authentication by themselves, which is error-prone, inconvenient and often insecure. Now the semi-automated Registration Service takes care of the distribution of identities. Besides being more secure, our approach is also user-friendlier than the quite cumbersome manual deployment of passwords to devices/services as it is typically done in home networks today.

Key Findings and Contributions

▷	The generation and certification of asymmetric key pairs needed by the identification/authentication scheme presented in this thesis is a process that overburdens most inexperienced users.
▷	Assistance functionalities are needed that automate this difficult to understand and perform process. Otherwise, the identification/authentication scheme cannot be used in unmanaged networks.
C5.1	The **Registration metaphor** is easy to understand even by inexperienced users.
C5.2	The **Registration Service** automates the creation, certification and secure distribution of keying material to entities that belong to a Domain and hides all complexity and technical details from the user.

6. User-Friendly and Secure Trust Establishment

The creation of the identification and authentication scheme presented in Chapter 4 laid the cornerstone for access control in unmanaged networks, such as home networks. The subsequent introduction of the Registration Service in Chapter 5 enabled the different Domain Owners to manage the identities of entities that belong to their Domain effortlessly and securely.

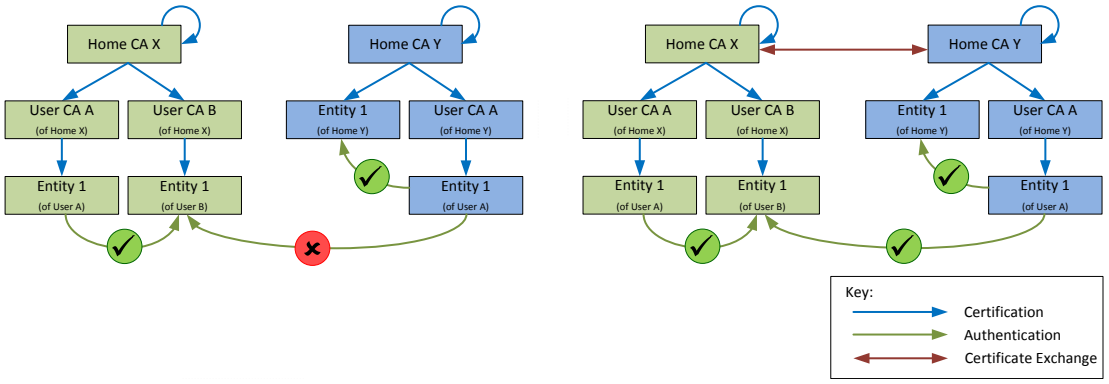
The result of both presented techniques is that a certificate chain (or trust chain) anchored in the Domain CA of the Home Domain to all entities that belong to this home is established. This makes the identification and authentication of all entities that belong to the *same* home possible. Thereby it does not matter if an entity is directly registered to the Home Domain or to a User Domain that belongs to this home, see Figure 6.1(a). Unfortunately, the trust chain stops at the border of the home. So when a *foreign* entity, an entity registered to a Domain that belongs to a *different* home, wants to access a service in the local home the authentication of this entity is impossible and as a result the service access is denied, see Figure 6.1(a). The reason for this problem is obviously that an unknown and untrusted Domain CA certified the foreign entity.

However, being able to authenticate foreign entities is a required feature of our access control system in order to allow the sharing of resources across home networks. This requirement has been specified as Requirement RD "Access Control across Domains" in Section 2.4. Requirement RE "User Friendliness and Security per Default" also influences the solution to this problem.

In this chapter we present our approach for establishing so called *Trust Relationships* between different Domains that do not belong to the same home. The establishment of Trust Relationships, which is called *Trust Exchange*, is based on the idea to exchange the certificates of Domain CAs between different Domains. This simple approach will enable the authentication of entities across homes and finally the sharing of resources across home networks, see Figure 6.1(b).

Chapter Structure

In the following Section 6.1 the approach to exchange trust between Domains that belong to different homes is discussed and requirements on the Trust Exchange protocols are defined. Section 6.2 introduces the first Trust Exchange protocol, which is executed in person



(a) No trust chain between homes. Authentication and service access is impossible.

(b) Trust chain established. Authentication and service access is enabled.

Figure 6.1: Establishing Trust between domains enables authentication and service Access.

between authorized representatives of a Domain. A second Trust Exchange protocol, which is executed over the Internet, is presented in Section 6.3. The Chapter is finally concluded in Section 6.4

Acknowledgments

This chapter contains results developed in the Diploma Thesis of Blaž Primz[1] [37], the Diploma Thesis of Gürcan Karakoc [42], and the Master's Thesis of Simon Mittelberger [38]. All works were assigned and supervised by the Author in the context of his doctorate.

6.1 Trust Exchange

As we have discussed in Section 2.2 we expect that users desire to share resources (services or data) hosted within their homes primarily with persons or groups of persons from their close social environment, e.g., family members living in other homes, friends, colleagues, etc. Furthermore we expect that typically only a relatively small number of "connections" to the world outside the own home is needed. Additionally, we have already argued in Section 4.2.2 that traditional PKIs and trust models (either X.509 or PGP/GPG) are not suitable and useful in our scenario. By these reasons we are convinced that the best way to establish Trust Relationships between Domains that belong to different homes is user-centric and based on social interactions between their owners.

Before we continue with this discussion, we want to define the central terms in this chapter: A *Trust Relationship* between two Domains A and B exists when Domain B trusts the CA of Domain A (and vice versa). In other words, certificates issued by Domain CA A are regarded valid in Domain B. This will finally enable the authentication of entities that belong to Domain A in Domain B. For this reason, all Domains that belong to the same home already share an implicit Trust Relationship. Consequently the establishment of a Trust Relationship between two Domains is called *Trust Exchange*.

6.1.1 Social Graph vs. Trust Relationships

In the following the relationship between the social graph of a user and the graph of Trust Relationships (trust graph) between this user's User Domain and other Home and User Domains is discussed in order to understand the properties of Trust Relationships better. An example that will support these considerations is given in Figure 6.2.

[1] Associated to the University of Ljubljana. Performed his Diploma Thesis at the Technische Universität München.

The user symbolized by the red circle is the "center" of this example. She belongs to three different communities, namely family, friends and colleagues, see social graph (top layer) of the figure. Between this user and all members of her social graph exists some sort of social relationship.

This user wants to share resources that belong to her personal User Domain only with selected members of her social graph (bottom layer). For this reason, the trust graph of this user will only contain a subset of members from her social graph. In the trust graph, these selected members are represented by their User Domains, which in turn belong to a Home Domain each.

This example demonstrates clearly that Trust Relationships between Domains may exist on different layers of the hierarchy of Domains in a home.

For instance, the user has two family members that live in a different home (Home D) than herself. This user wants to share resources with her family members. For this purpose it might be possible that she establishes two different Trust Relationships to each User Domain of the respective family members. But it is also possible to establish a *single* Trust Relationship between the Home Domains A and D. This will enable all entities that belong to the User Domains of Home A and D to authenticate each other. In this case, as all members of both homes have a quite strong social relationship and the chance is therefore high that they might wish to share services with each other, this approach is quite reasonable. In fact, establishing a Trust Relationship between both homes is more efficient and simpler than establishing several Trust Relationships between single User Domains.

In contrast, it is not reasonable to establish Trust Relationships between the user's Home Domain and Home Domains of friends or colleagues. The reason for this is that the user does not know any other member of these homes, which will also mean that there is most likely no reason to share services with them and finally to authenticate their entities.

Finally, also Trust Relationships across different Domain hierarchy layers are possible, which is not depicted in the figure. This might be needed to connect a families Home Domain with the User Domain of a family member that lives in a flat sharing community, for instance.

Please note: In environments that need more than two Domain hierarchy layers, like companies, the situation is almost the same. Depending on the scenario, establishing trust between Domains that reside on the same or a different hierarchy layer is needful to achieve the desired effect.

6.1.2 Approach

Comparable to the Registration Service that distributes identities within Domains, the establishment of Trust Relationships between Domains must be packed in an easy to understand concept and be implemented in software that guides the inexperienced users through the difficult to understand and perform steps. Additionally, as establishing trust is security relevant, this software must take care of the security of the process.

The entire Trust Exchange process between two Domains can be split into three steps.

1. **Identification/Authentication:** The Domains that want to establish trust need to mutually identify/authenticate each other.

2. **Certificate Exchange:** The Certificates of the Domain CAs that represent the Domains need to be exchanged.

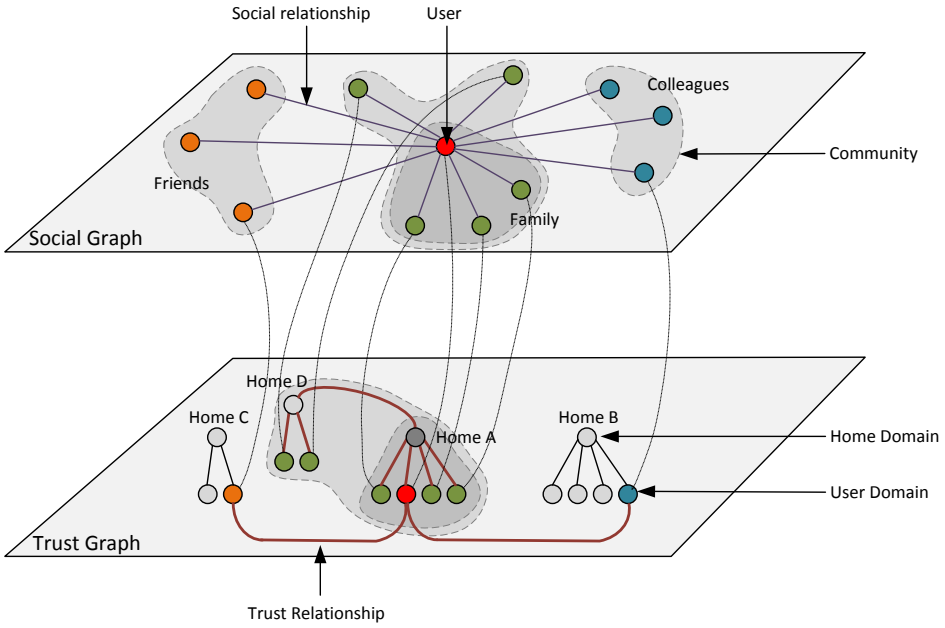


Figure 6.2: Social graph and trust graph of a User: Most Trust Relationships exist between User Domains (friends, colleagues). But also Trust Relationships between Home Domains are possible.

3. **Certificate Propagation:** Finally, the certificate received from the Partner Domain needs to be made available to all entities that belong to the local Domain. Depending on the context this means that

   (a) the received certificate is made available to entities that belong to the own User Domain.

   (b) the received certificate is made available to all entities that belong to the home.

Figure 6.3 gives an example. If Trust Exchange is performed between User Domains, e.g., between Domain B and C, the received certificates only need to be propagated to the private entities that belong to these Domains (case a). In case trust is established between Home Domains, e.g. Domain X and Y, the certificates need to be made available to public entities that belong to the Home Domains and to private entities of User Domains (case b).

The last step, the propagation of trusted Domain certificates to entities, is quite easy to automate. For this purpose only some sort of synchronization mechanism is needed. For instance, received certificates might be stored in a database of trusted Domain CA certificates first. For instance, once a day, all entities that belong to the Domain synchronize their own, local database of trusted Domain CA certificates (case a). The propagation of certificates exchanged between Home Domains to User Domains (case b) can be solved in the same fashion. A difference might be that the owner of a User Domain wants to acknowledge the import of a certificate into her own database first.

More interesting than the propagation of received certificates inside the home are the first two steps, namely identification/authentication and the certificate exchange itself. In Sections 6.2 and 6.3 we present two mechanisms that will perform these tasks. Before both mechanisms are detailed, various requirements need to be defined and two terms introduced.

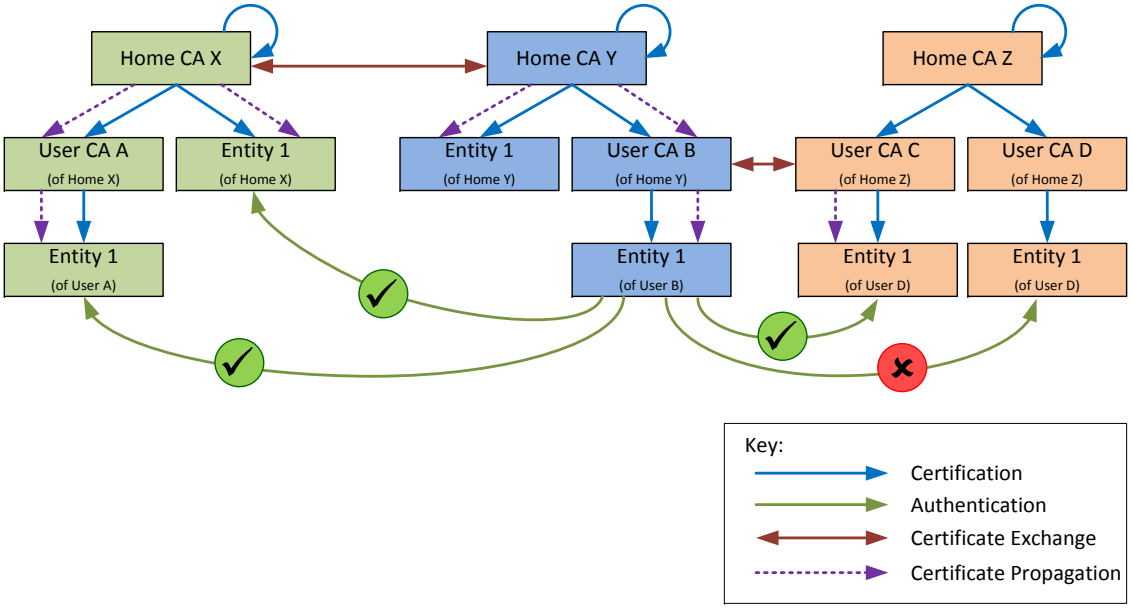


Figure 6.3: Certificate Exchange and Propagation

6.1.3 Identification and Reputation Level

In the following, two variables are introduced, which allow measuring the strength of a Trust Relationship between two Domains:

The *Identification Level (IL)* is a property of a Domain certificate. It is a numerical value in the range from 0 to 10 that measures the confidence a Domain A can have that a certain Domain certificate belongs to another Domain B. Either the Domain Owner (or a representative) that performs a Personal Trust Exchange or the Internet Trust Exchange service assigns the IL. ILs are "asymmetric", i.e., the Domains that share a Trust Relationship with each other may have assigned *different* ILs to the other Domain.

The *Reputation Level (RL)* is a property of a Domain Owner. It is a numerical value in the range from 0 to 10 that expresses the expectation of a Domain Owner A how much care a Domain Owner B will take when she is performing a Trust Exchange with a Domain Owner C, i.e., how reliable Domain Owner B will check the Identity of Domain Owner C and then assign an appropriate IL to the received certificate.

IL and RL ratings are needed by the Trust Metric we introduce later in Section 6.3.3 to calculate the IL of a Trust Relationship established over the Internet.

6.1.4 Requirements

The requirements on the implementations of the Trust Exchange can be grouped into requirements related to security and user-friendliness.

Security of Trust Exchange

Establishing trust to the outside of the home is a critical task. When the security of the process is not guaranteed, unintended Trust Relationships might be established. In the worst case the falsely created Trust Relationship is abused to access services that control critical physical elements of the home, for instance.

- **R1: Secure Identification/Authentication of the Trust Exchange Partners:** The aim of a Trust Exchange is to receive the certificate of a specific Partner Domain. This aim is only fulfilled if the Partner Domain could be identified/authenticated and if it can be ensured that the received certificate belongs to this Domain.

- **R2: Rating of Certificates:** In some situations it is not possible to undoubtedly identify/authenticate the Partner Domain. Therefore, the Trust Exchange must assign an Identification Level to a certificate received from a Partner Domain, which expresses the strength of the established Trust Relationship.
- **R3: Security:** All precautions need to be taken that a third party is unable to interfere with the Trust Exchange between two Domains, e.g., to trick one or both partner to trust a Domain certificate owned by the third party.
- **R4: User-Centrism** The Trust Exchange must be user-centric, i.e., may not depend on external (central) services such as public CAs whose trustworthiness cannot be verified or key servers that create privacy problems.
- **R5: Privacy Preserving:** Existing Trust Relationships between Domains must be kept secret, i.e. no third party should be able to automatically enumerate Trust Relationships of a Domain, which are a subset of the Domain Owner's social graph.

User-Friendliness of Trust Exchange

Performing a Trust Exchange between Domains must be an easy to perform, to understand and user-friendly process. Therefore following requirements must be fulfilled:

- **R6: High Degree of Automation:** The Trust Exchange must hide all difficult to understand technical details and run mostly automatically.
- **R7: Owner Consent:** No Trust Relationship may be established without the Domain Owner's permission.
- **R8: Low Interference:** The amount of interactions between the Trust Exchange mechanism and the human user that is involved in the process must be kept at a minimum to avoid repeated disturbance and prevent annoyances.
- **R9: High responsiveness:** Long waiting times until the Trust Exchange has finished need to be avoided, if possible. It is not expected that the process is computationally expensive, so the responsiveness mostly depends on the protocol design and response times of involved human users.
- **R10: Versatility:** The Trust Exchange process must be generic, i.e. the same mechanism must be able to establish trust between Domains that belong to the same hierarchy layer (Home Domain to Home Domain, User Domain to User Domain) or between mixed Domain types (Home Domain to User Domain).

6.2 Personal Trust Exchange

This section introduces a Trust Exchange protocol that is executed between the devices of two Domain Owners or their representatives during a personal meeting.

6.2.1 Approach and Basic Architecture

The most straightforward and natural way to perform the Trust Exchange between two Domains is during a personal meeting of two Domain members, typically the owners of the involved Domains. This idea corresponds to the direct trust model discussed in Section 3.3.1.

The so-called *Personal Trust Exchange* is assisted by easy to use software components that implement the needed functionality and guarantee the required security of the process.

These applications can be installed on devices that belong to Domains, for instance smart phones or notebooks.

As communication medium between devices we decided to use *radio frequency communication (RFCOMM)* over Bluetooth, which provides a reliable data stream comparable to TCP over IP. Alternatively, an ad-hoc WLAN or NFC could be used as well. The protocol we have designed is agnostic of the used communication medium as it implements all needed security mechanisms itself.

After the Trust Exchange between devices is performed, the received certificate of the Partner Domain needs to be imported into the own Domain. For this purpose the Guided Security Management System of a Domain running on the Domain Server offers a component called *Import Service* that receives certificates of Partner Domains and stores them in a database from where they can be synchronized to entities that belong to the Domain. In case the user that performed the Trust Exchange and the Domain Owner are the same person, this import is performed without any further inquiry. But when a resident wants to import a certificate to a Domain she does not own, e.g., a resident wants to import the certificate of another Domain into the own Home Domain, the owner of the Home Domain must be asked for permission first.

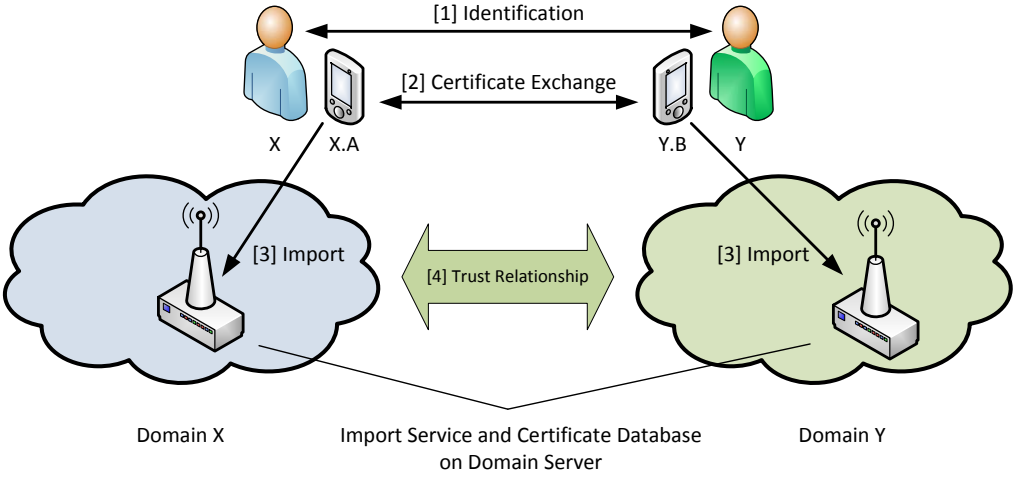


Figure 6.4: Personal Trust Exchange (architectural overview): Step 1, the participating parties identify each other. Thereafter, their Domain certificates are exchanged (Step 2). Finally the received certificate is imported into the own Home or User Domain (Step 3).

6.2.2 Protocol

Phase One: Discovery of Available Trust Exchange Partners

In phase one, see Figure 6.5, the physical Bluetooth connection between the devices of the exchange partners is established.

Domain Member X starts the Trust Exchange in Step 1. Her device X.A first discovers possible partner devices in Trust Exchange mode (Step 2) and displays the results to X (Step 3). Typically only one or few devices should be found. The Domain Member X selects the desired partner device Y.B (Step 4). Subsequently device X.A connects to Y.B and requests the Trust Exchange (Step 5). Device Y.B asks its user for permission to participate in the Trust Exchange with Domain X. Finally user Y accepts the Trust Exchange request (Step 6).

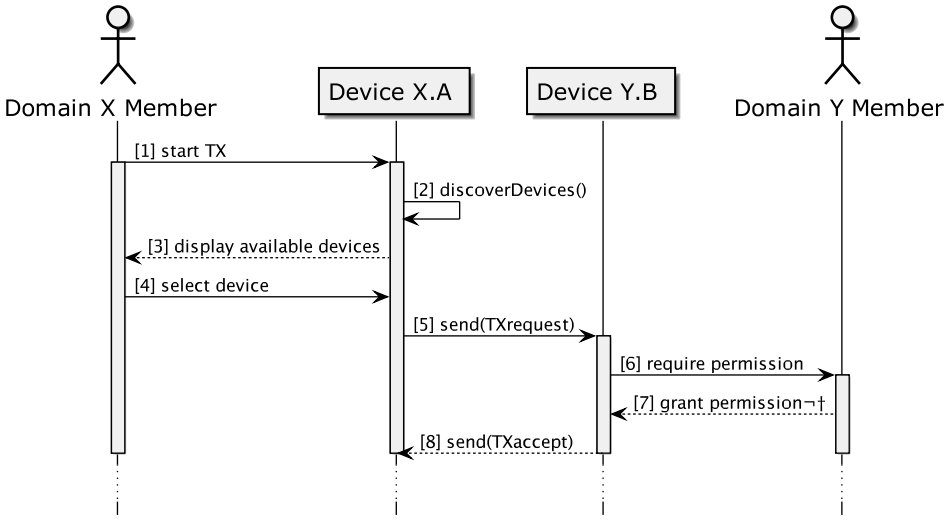


Figure 6.5: Personal Trust Exchange (details), phase 1: Discovery of available Trust Exchange partners.

Phase Two: Establishment of a Secure Channel between Devices

Phase two, depicted in Figure 6.6, exchanges and authenticates a shared secret session key between both involved devices X.A and Y.B over the insecure RFCOMM connection. The same approach as used in the Registration Protocol could be used here as well. In this case we decided to use a slight modification that offers more convenience to users.

The involved devices X.A and Y.B first agree automatically on the session key using the Diffie-Hellman key agreement protocol [43] (Steps 1 - 8). As the protocol is prone to Man-in-the-Middle attacks, the established session key must be authenticated before it can be used to encrypt the subsequent messages of the Trust Exchange protocol[2]. As the participating devices share no trusted cryptographic keying material yet, no fully automated protocol can be implemented. This issue can be solved easily by involving the users and ask them to verify the exchanged session key.

For this purpose the following mechanism was implemented. After the session key is generated on both devices (Steps 7 and 8), a cryptographic hash digest of the session key is generated on each side. The hash digest is then trimmed to the first 16 bits (Steps 9 and 10). The result of this procedure is a so-called *Short Authentication String (SAS)*, which is then displayed to the users in hexadecimal notation alongside the request to compare the values displayed on both devices (Steps 11 and 12). If the SASs displayed on both devices are equal, the same session key was established on both sides, which proves that no attacker was able to interfere. In this case, the users advise their devices to continue with the Trust Exchange (Steps 13 - 15 and 18 - 20).

After the verification of the session key, the devices request their users to provide meta-information about the other party. The Identification and Reputation Level (IL, RL) of the exchange partner (Steps 16 - 17 and 21 - 22), a human readable identifier (hID) and a community (family, friends, colleagues, etc.) this Domain belongs to need to be provided.

To facilitate the rating of IL and RL, the devices display various options that can be selected by the users. Displayed options include, for instance, “I know the exchange partner for a long time and am convinced to know her correct identity” or “I verified the exchange partner’s identity using her identity card”. Both options will be mapped to a numerical IL,

[2] A successful Man-in-the-Middle attack on the Diffie-Hellman key agreement would enable an attacker to interfere with the subsequent certificate exchange.

in this case 10, which reflects the undoubtedly identification of the exchange partner. The system also displays options the user might select when she is unsure about the partner's identity, e.g., "I only know the exchange partner for a short time and am unsure about her identity". In such a case the IL will be mapped to a low IL, e.g., 3.

For rating the exchange partner's Reputation Level similar help is provided. An option displayed by a device that results in a high RL is "I am convinced that the exchange partner performs Trust Exchanges with great care". This option will be mapped to a high RL of 10. A negative example is "I have strong doubts that the exchange partner performs Trust Exchanges with great care". Again, a mapping of the selection to a low numerical value is done automatically.

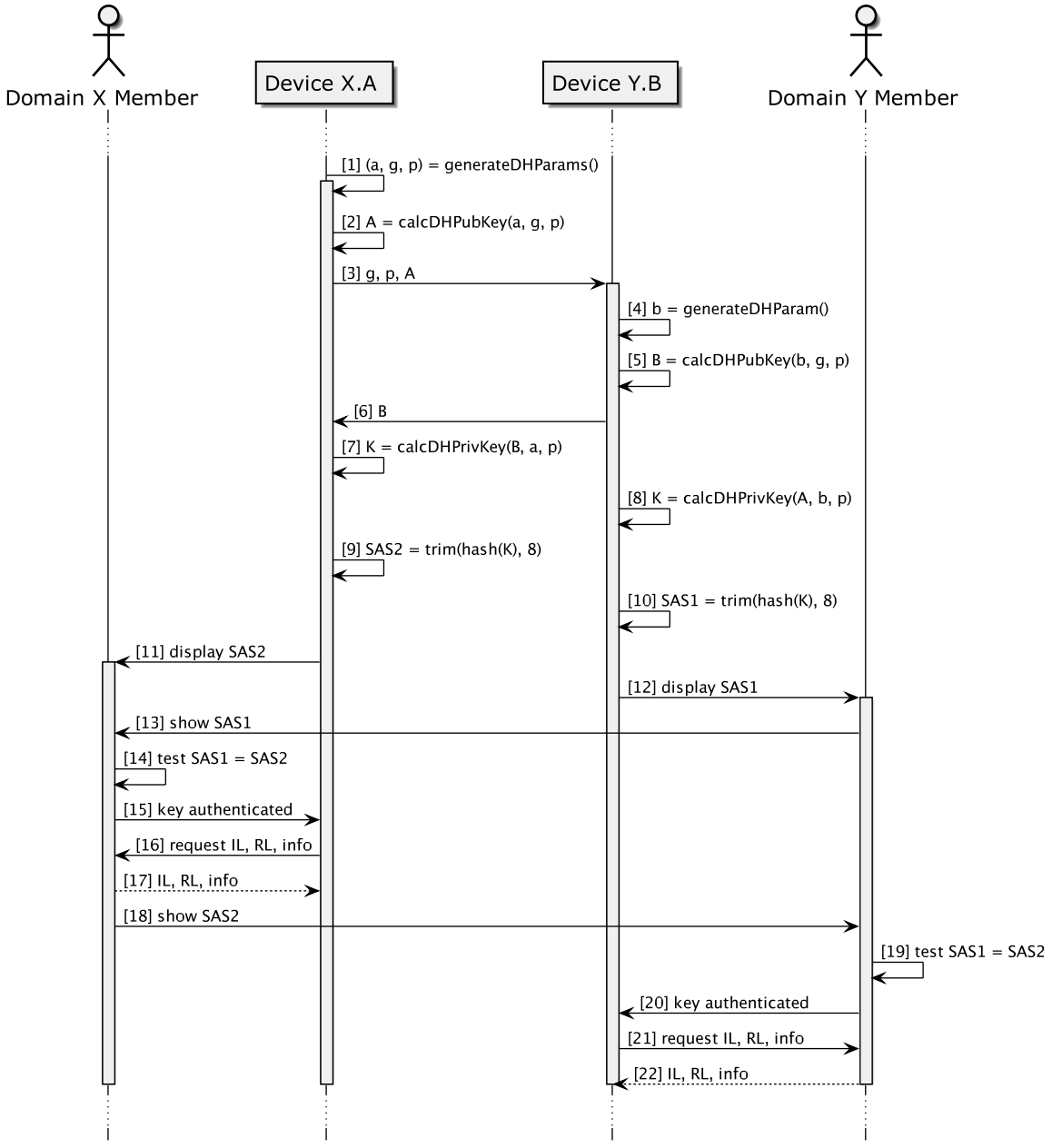


Figure 6.6: Personal Trust Exchange (details), phase 2: Establishment of a Secure Channel between Devices.

Please note: the collection of IL, RL and community name are actually not necessary for the Personal Trust Exchange protocol. This information is needed by the Internet Trust Exchange described in Section 6.3.

Phase Three: Certificate Exchange and Verification

After finishing all preparations, phase three, the actual certificate exchange, is performed, see Figure 6.7. All messages exchanged between the devices are encrypted using the previously exchanged session key.

Device X.A sends its own device certificate (certX.A) and the certificate of its Domain CA (certX) to the other device in Step 1. Y.B first tests if the received device certificate was signed by the Domain CA (Step 2). For this purpose the signature of the device certificate is verified using the public key taken from the received Domain certificate. If this test succeeds, Y.B is confident that the presented device certificate belongs to the expected Partner Domain and sends its own certificates to X.A (Step 3). X.A performs the same test for the received certificates than Y.B did before (Step 4). This first test is necessary to continue with the Trust Exchange but not yet sufficient.

The second test verifies if the other device possesses the private key that belongs to the just presented device certificate. For this purpose X.A computes a random nonce (rand_1) and sends this value to Y.B (Step 5). Y.B participates in a simple challenge/response protocol that proves the possession of the corresponding private key and encrypts rand_1 using her private key (Step 6). The resulting value (resp_1) is send back to X.A (Step 7). X.A verifies the response using the public key of Y.B taken from the previously received device certificate. If the received response matches the sent challenge, X.A is confident that Y.B is the legitimate owner of the device certificate and also a valid member of the Partner Domain (Step 8 - 9). This confidence is necessary in order to accept the received Domain CA's certificate (Step 10).

The same challenge/response protocol is used by Y.B to test if X.A possesses the private key belonging to the presented device certificate (Steps 11 - 16).

The actual certificate exchange between both Partner Domains is now finished.

Phase Four: Certificate Import

After a device performed one or several Trust Exchanges with Partner Domains, the received certificates and assigned meta-information (IL, RL, human readable identity) need to be imported into the own Domain, see Figure 6.8.

For this purpose, device X.A sends all previously received information to the Import Service (Step 1 - 2) of her own Domain. In case the *Domain Owner* performed Trust Exchange herself, no further inquiry whether to import the certificate needs to be made by the Import Service. In case another resident of the home wants to import a certificate, the Import Service first asks the Domain Owner for permission (Step 3 - 4). Alongside with giving the permission, the Domain Owner can modify meta-information assigned by the resident to the received certificate, if needed. Finally, if the certificate should be imported, the Import Service stores the received information in the Domain's database for trusted certificates (Step 5).

6.2.3 Discussion and Evaluation

6.2.3.1 Fulfillment of Requirements

The above-presented protocol meets all requirements we have defined in Section 6.1.4.

As the process is performed during a personal meeting it is easy for the exchange partners to identify each other (Requirement R1). This is because the partners already know each other (which we strongly assume) or because they were able to verify each other's identities

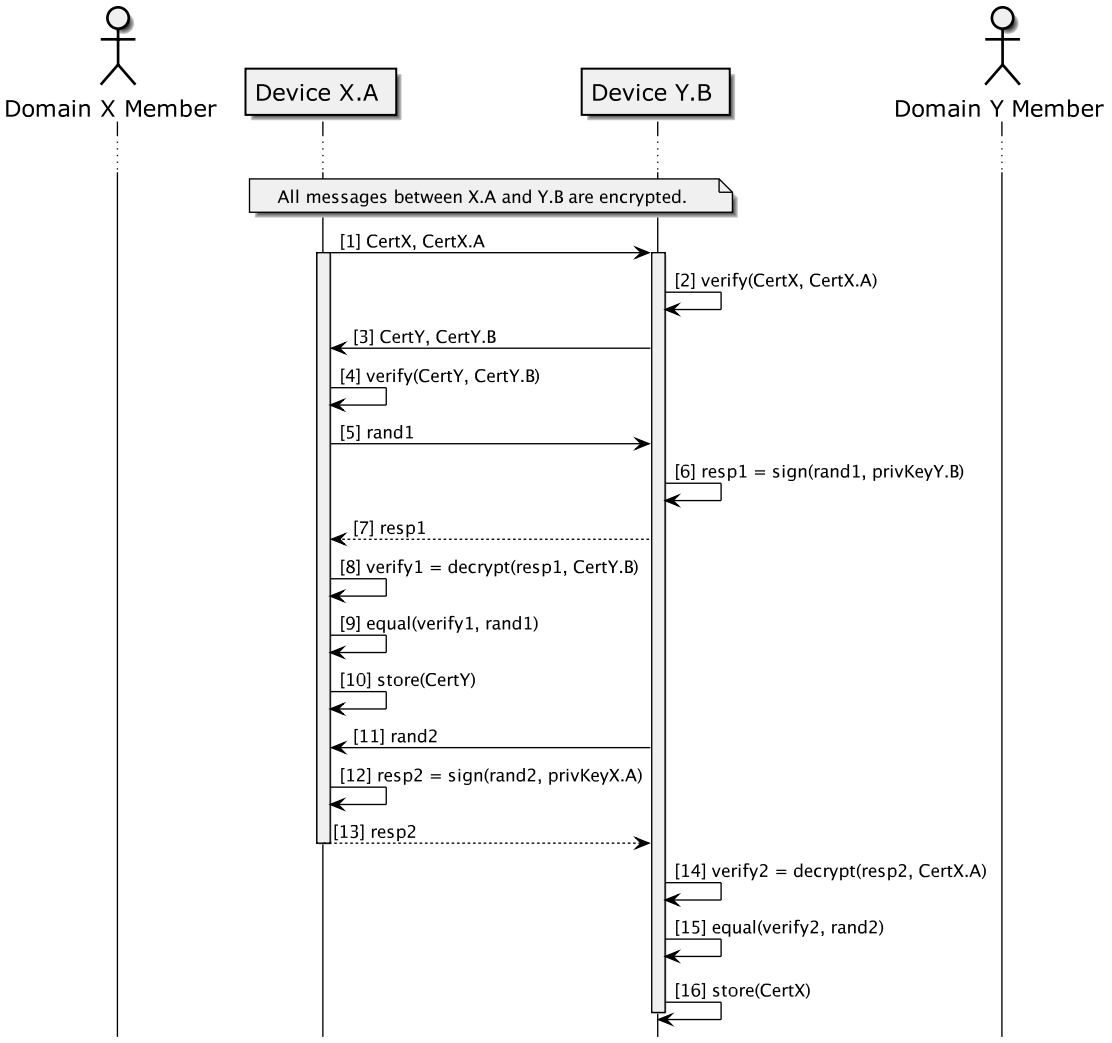


Figure 6.7: Personal Trust Exchange (details), phase 3: Exchange of certificates with additional tests.

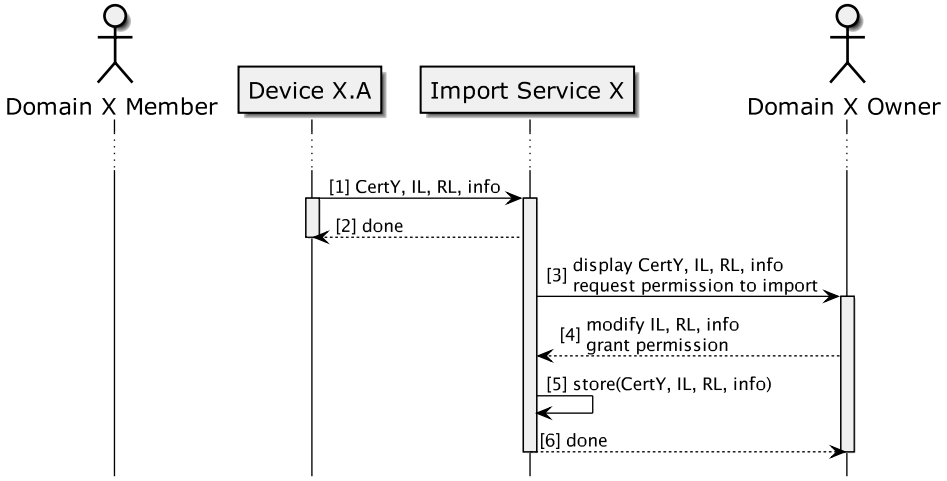


Figure 6.8: Personal Trust Exchange (details), phase 4: Import of a received certificate and meta-information into a Domain.

using an official identity card. Furthermore, various tests are applied to prove that the partner device is indeed a member of the partner Domain.

The certificates are exchanged over a channel secured by using state of the art cryptography and an authenticated secret session key. For this reason we can exclude that an attacker is able to impersonate another user, i.e., interfere with the certificate exchange and inject faked certificates (R3). The authentication of the session key using the SAS is sufficient, as even a very short SAS makes it difficult for the attacker to remain undetected. *"A 16-bit SAS [...] provides the attacker only one chance out of 65536 of not being detected."* [44] For the users this step is effortless to perform and no major inconvenience is caused, as they only need to compare four hexadecimal characters. An alternative to the SAS might be that the devices compute a so-called *Identicon*[3] based on the hash value of the session key. Identicons are visual representations of abstract data, such as of hash values. The users can compare this visual representation even easier than testing the equality of strings.

As the process obviously does also not depend on any external party (R4) the privacy of the involved Domains remains unharmed (R5). Finally, the mechanism collects all required meta-information about the Trust Exchange partner during the process (R2).

The Trust Exchange is quick and easy to perform and offers as much automation as possible (R6). A higher degree of automation would result in losing control over the own Domain, as the system might automatically perform actions that are not desired (R7) by the Domain Owner.

A task that cannot be automated completely is the collection of meta-information (IL, RL, human readable identifier) about the Partner Domain, which must be done by the involved human users. As the Internet Trust Exchange needs this information, it is impossible to waive this step. Nevertheless, the devices assist their users so that no major annoyance needs to be expected (R2).

The amount of interaction between the system and the user is limited. Less interaction would be possible with a higher degree of automation, but, as described above, this will have negative influence on the controllability of the system and on the collection of meta-information (R8). Furthermore, the protocol was designed to keep time-wise dependencies between user inputs at a minimum (R9).

Finally, the mechanism can be used in multiple scenarios, i.e., to establish trust between Home Domains, Home and User Domains, User Domains and also in a completely different scenario than the home network (R10).

6.2.3.2 Benefits and Drawbacks

The strength of the mechanism presented is that the exchange of certificates is performed "directly" and in person during a meeting. This direct approach provides a perfect identification of the exchange partner. Therefore, a high IL can be typically assigned to the received certificate. Strong Trust Relationships between Domains, i.e., Trust Relationships that have a very high IL, are essential for the Internet Trust Exchange described in the subsequent section.

This strength of the Personal Trust Exchange is also a major weakness, as the process requires that the exchange partners meet personally. Nevertheless, as no other Trust Exchange mechanism will be able to establish stronger Trust Relationship between Domains, the Personal Trust Exchange should be regarded as the default way to establish Trust between Domains.

[3] For instance, http://identicons.sourceforge.net/

6.3 Internet Trust Exchange

Exchanging trust using the Personal Trust Exchange described above is only possible if members of both Domains are able to meet in person. In many situations, e.g., when families live abroad, a personal meeting is impossible. So a mechanism is needed that functions without a personal meeting.

The simplest approach would be to manually exchange the certificates of Domains, e.g., by sending the own certificate to the Partner Domain in an email. But as email communication is insecure, an attacker might interfere and exchange the certificates. For this reason an out-of-band authentication mechanism for the received certificate is required. The users, for instance, could verify the received certificate by comparing its hash digest via telephone. This process implicitly identifies the other Domain member (assumed both persons are able to recognize each other voices, which is a fair assumption) and also authenticates the received certificate. Nevertheless, the described approach violates Requirements R4 (high degree of automation) and R6 (low degree of interaction) we defined above.

Another approach would be to publish the certificate of the own Domain on a personal social network profile, e.g., on Facebook, Google Plus, Xing, etc. This approach seems to be quite appropriate on first sight. The friends of the user that are able to access the social network profile are also able to obtain the certificate from there. This approach would also be quite secure and reliable, provided that the social network profile of the user is authentic and that the user protects her profile with a strong password. This will prevent that third parties can replace the published certificate with a faked version. In fact, we have designed and implemented a Trust Exchange protocol based on Facebook. But we also see major drawbacks. First, there is a high dependence on the used social network service, which violates Requirement R4. Furthermore, not every Domain Owner is or wants to be part of a social network service and not all Domain Owners will be part of the *same* social network service. For this reason we do not detail this approach here, but the interested reader may find further information, protocol flows and implementational details in [42].

A third option is to use an approach comparable to PGP/GPG. A Domain Owner A might publish her Domain certificate (certA) on a public key server or another publicly accessible storage mechanism. This kind of mechanism might be implemented based on a Peer-to-Peer network and use a DHT to retrieve needed certificates. Other Domains that have already verified certA may create and publish signed data structures that confirm the authenticity of certA, which is quite comparable to cross signing. Finally, another user B that wants to connect her Domain with Domain A might use this information to establish trust in certA. The biggest drawback of this approach is that social network structures are publicly revealed. For this reason the social graph of a user can be reconstructed. In our opinion this is a major problem and we are convinced that the privacy of the users should be protected.

6.3.1 Approach and Basic Architecture

Our approach for a secure, privacy conserving and reliable mechanism to exchange trust between Domains over the Internet is based on the idea to propagate existing Trust Relationships between Domains in a controlled manner. For this purpose the so-called *Internet Trust Exchange* protocol was designed.

In this protocol, Domains, more precisely, their owners, can be divided into three different roles: The Domain Owner that wants to establish a new Trust Relationship between her and a friend's Domain is called *Requester*. The friend that is invited to join the new Trust Relationship is called *Invitee*. Finally, common friends that assist Requester and

Invitee to establish the Trust Relationship are referred to as *Counselors*. Correspondingly, the Domains that belong to these persons will be called Requester, Invitee or Counselor Domain.

The service that implements the Internet Trust Exchange protocol is called *Exchange Service* and is, as the Registration Service, a component of the Guided Security Management System of a Domain hosted on the Domain Server. The Exchange Service processes incoming requests, interacts with the Domain Owner using an easy to use graphical user interface (GUI), and finally responds to requests on behalf of the Domain Owner. Figure 6.9 provides an overview of the named roles and services.

The Exchange Services of Domains involved in the Internet Trust Exchange need to be addressable from the Internet. For this purpose we use the already introduced idea to create a Peer-to-Peer network of Domain Servers that constitute a DHT. This DHT stores key/value pairs (cryptographic ID of Domain/IP of Domain), which finally allows the look up of the IP address, which is currently assigned to a specific Domain's Exchange Service, see Section 4.4.

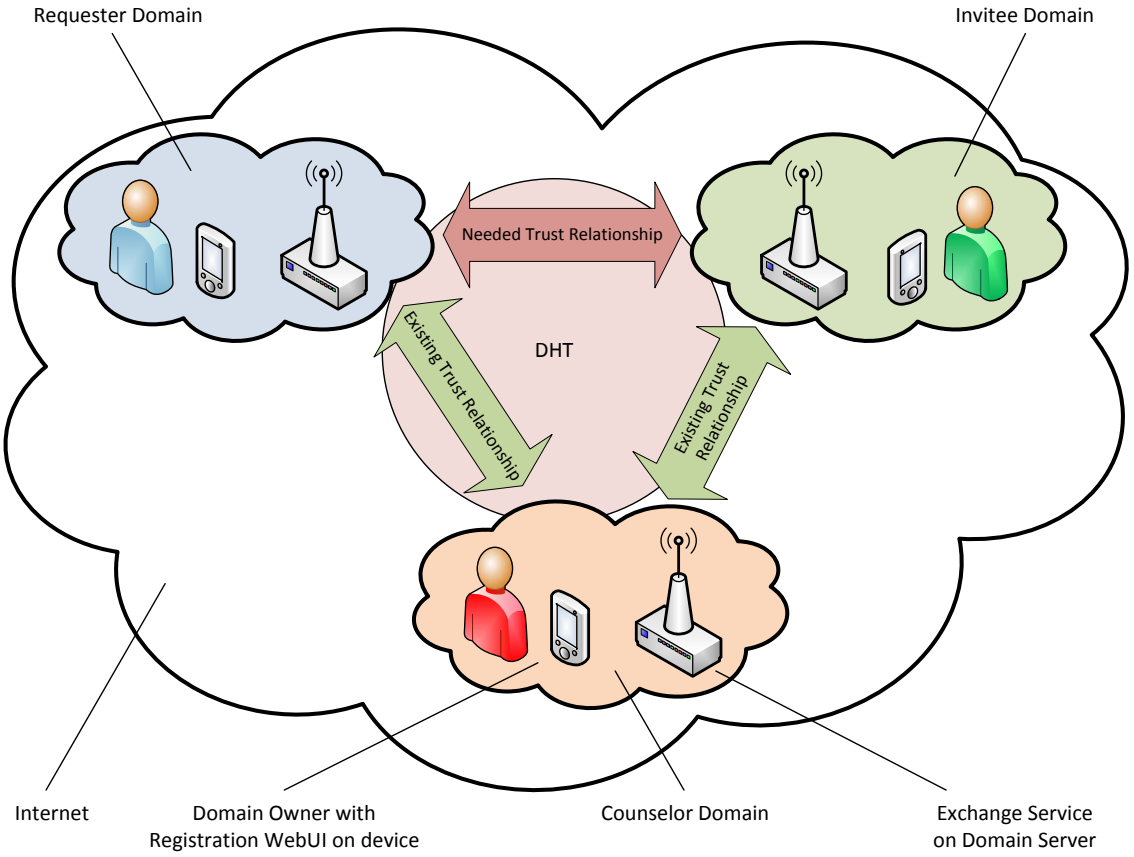


Figure 6.9: Indirect Trust Exchange (architectural overview)

Figure 6.10 depicts the basic ideas and mechanisms of the Internet Trust Exchange, which will be detailed in the following section.

1. The owner of Domain R, the Requester, wants to establish a Trust Relationship to her friend, the Invitee. Fortunately, Trust Relationships exist already between herself and a suitable Counselor Domain C and between Domain C and the Invitee Domain I.

2. The Requester initiates the protocol by sending the human readable identity of the Invitee (name, e-mail address, etc.) to Domain C. This message is called *Counsel*

Request. Domain C answers the request with a *Counsel Response*, which contains the cryptographic identity cID_I of the Invitee Domain and the Invitee Domain's certificate certI.

3. Domain R uses the received cID_I to look up the IP address of Domain I, which is needed to send an *Exchange Request* to Domain I. The subsequently sent Exchange Request contains the cryptographic identity cID_C of the involved Counselor Domain, the human readable identity of the Requester (name, e-mail address, etc.) and Domain R's certificate certR.

4. At this time Domain I is unable to decide if the Exchange Request was indeed sent by her friend, the Requester, i.e., if certR is indeed the certificate of Domain R. Domain I now uses the existing Trust Relationship to Domain C and asks Domain C to authenticate the Exchange Request and certR for her. For this purpose an *Authentication Request* containing the just received information is sent to Domain C. Domain C verifies the Exchange Request and answers with a positive *Authentication Response*.

5. After receiving the positive Authentication Response, Domain I can be confident that the Requester sent the Exchange Request and that certR belongs to Domain R. Finally, Domain I agrees to participate in the Trust Relationship by sending an *Exchange Response* to Domain R.

6. The protocol is finished and a new Trust Relationship is established between Domains R and I.

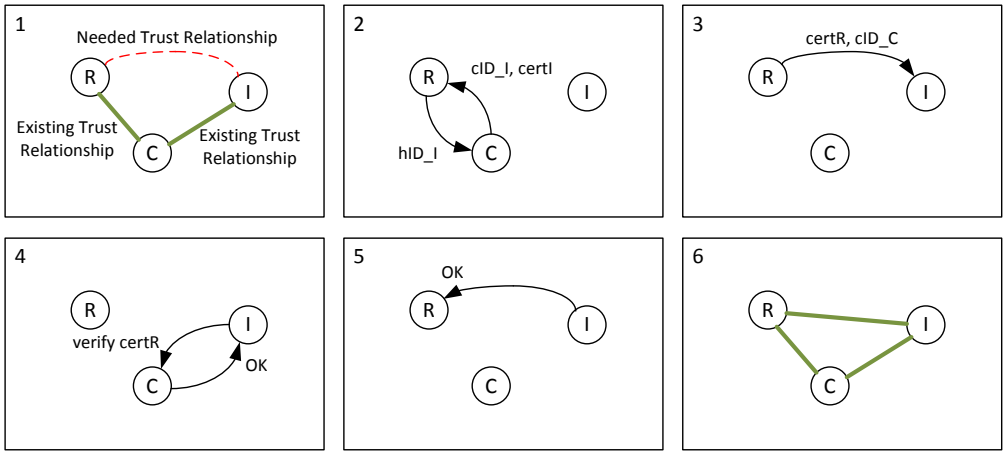


Figure 6.10: Internet Trust Exchange (approach)

6.3.2 Protocol

Phase One: Sending Counsel Requests to Suitable Counselors

The first phase of the Internet Trust Exchange protocol, depicted in Figure 6.11, initiates the process. The Requester specifies to which Invitee Domain she wants to establish a Trust Relationship and which Counselor Domains should be used for this purpose.

The careful selection of Counselor Domains is needful due to multiple reasons. First, a Counsel Request sent to a Counselor Domain will reveal the social contact between Requester and Invitee. For this reason, the Requester might not want that *all* Domains she has established a Trust Relationship with (trusted Domains) are requested in order to protect her own privacy. Second, sending Counsel Request to *all* trusted Domains is

not useful when we assume the existence of distinct social communities as described in Section 6.1.1. For instance, it is not useful to send a Counsel Request to the Exchange Service of a friend's Domain when trust should be established to a colleague's Domain. Finally, a careful selection of Counselors helps to reduce the risk of performing a Trust Exchange with an unintended Domain. For instance, the Requester might know two John Smiths. By sending the Counsel Request to Domains that belong to members of the community the intended John Smith is part of will effectively reduce the chance that trust is accidentally established to the other John Smith's Domain.

In the following we explain the first phase of the protocol in detail.

In Step 1 the Requester starts the Trust Exchange. She provides the human readable identifier (hID_I) of the Invitee to her Exchange Service. Furthermore, the Requester names one or several social communities (Co) the Invitee belongs to.

The local Exchange Service selects suitable Counselors from the list of already established Trust Relationships, i.e., Counselors that belong to the same community as the Invitee, and sorts these Counselors according to their Trustworthiness Level (Step 2). Finally the result is displayed to the Requester (Step 3).

The Requester selects one or several proposed Counselors that should receive the Counsel Request (Step 4). Now the local Exchange Service computes a random number that identifies this Trust Exchange session (txID) (Step 5) and stores all parameters that belong to this session in its database (Step 6).

The local Exchange Service now performs the IP lockup of the desired Counselor Domain and performs a TLS handshake with mutual authentication with the Counselor's Exchange Service (Step 7) in order to protect the security and privacy of the subsequent communication. The mutual authentication of this connection is possible, as Requester and Counselor Domain already share a Trust Relationship. The local Exchange Service now sends a Counsel Request, which contains the human readable identifier of the Invitee (hID_I) and the Trust Exchange ID (txID) to the Counselor Domain's Exchange Service (Step 8).

The Counsel Request is received and stored by the Counselor Domain's Exchange Service (Step 9) and an acknowledgment is sent back to the Requester Domain (Steps 10 - 11).

Please note: If the Requester has selected several Counselors in Step 4, Counsel Requests will be send to several Counselor Domains, which is not shown in the figure.

Phase Two: Counsel Response

Phase two of the protocol is depicted in Figure 6.12. The Counselor decides whether she wants to reply to a previously received Counsel Request or not. This permission is required because sending a Counsel Response to the Requester will reveal the existence of a Trust Relationship between Counselor and Invitee. Various reasons exist why a Counselor might not want to reveal the existence of this social relationship and therefore does not want to participate in the protocol. Additionally, involving the Counselor is needful in case a Counsel Request is ambiguous. A reason for this might be when the Requester has specified a "fuzzy" human readable identifier of the Invitee, e.g., the Invitee's first name, or mistyped a name. In such a situation the Counsel Request cannot be processed automatically by the Exchange Service, but the Counselor might still understand the request and propose the correct Domain.

The Counsel Reply, which is sent from the Counselor Domain's Exchange Service to the Requester Domain's Exchange Service, contains information needed to securely connect to the Invitee's Domain, namely the cryptographic identity (cID_I) and the certificate of the

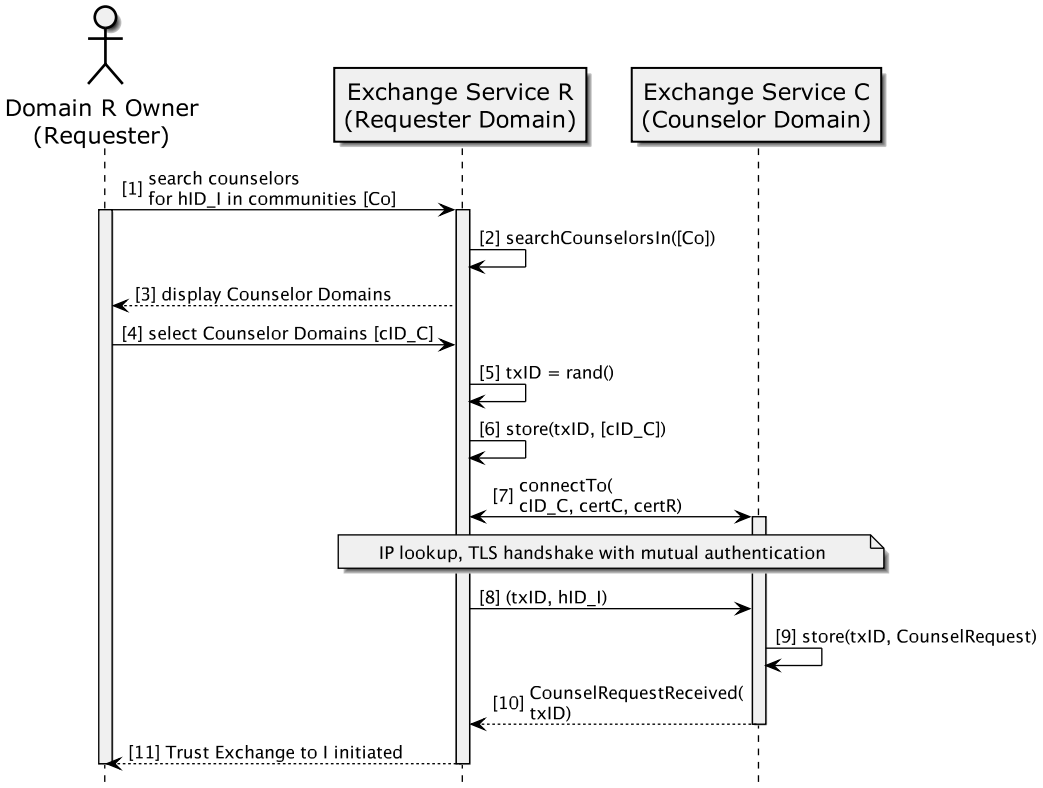


Figure 6.11: Internet Trust Exchange (details), phase 1: Specification of Invitee and Counselors and sending of the Counsel Request.

Invitee Domain (cert_I). Other meta-information, such as the Identification Level of the Invitee Domain (IL_I) are also contained. The IL is important for the Requester Domain's Exchange Service to compute the quality of the proposed information using the Trust Metric explained in Section 6.3.3.

The details of the second phase of the Trust Exchange are explained in the following:

At some point in time, the Counselor decides to handle the Counsel Requests stored by her Exchange Service (Step 1). For simplicity and brevity we assume that only one Counsel Request is stored at this time.

Before displaying the Counsel Request, the Exchange Service searches for the Domain that should be proposed to the Requester (Step 2). Typically only one Domain should be eligible, but in case the human readable identity contained in the Counsel Request was ambiguous, several Domains could be found. The found Domains and the corresponding Counsel Request are then displayed to the Counselor in Step 3. The Counselor now decides if she wants to respond to the Counsel Request at all. If yes, the Counselor selects one or several Domains found by the Exchange Service and finally gives her permission to the Exchange Service to send the Counsel Response (Step 4).

The Counselor Domain's Exchange Service establishes a secure channel to the Requester's Domain's Exchange Service (Step 5) and sends the Counsel Response (Step 6). The Counsel Response contains the cryptographic identity (cID_I), the certificate (certI) and the Identification Level (IL_I) of all proposed Domains and txID, which is used to assign the answer to the Trust Exchange session.

The Requester's Exchange Service receives, stores (Step 7) and acknowledges (Steps 8 - 9) the receipt of the Counsel Response.

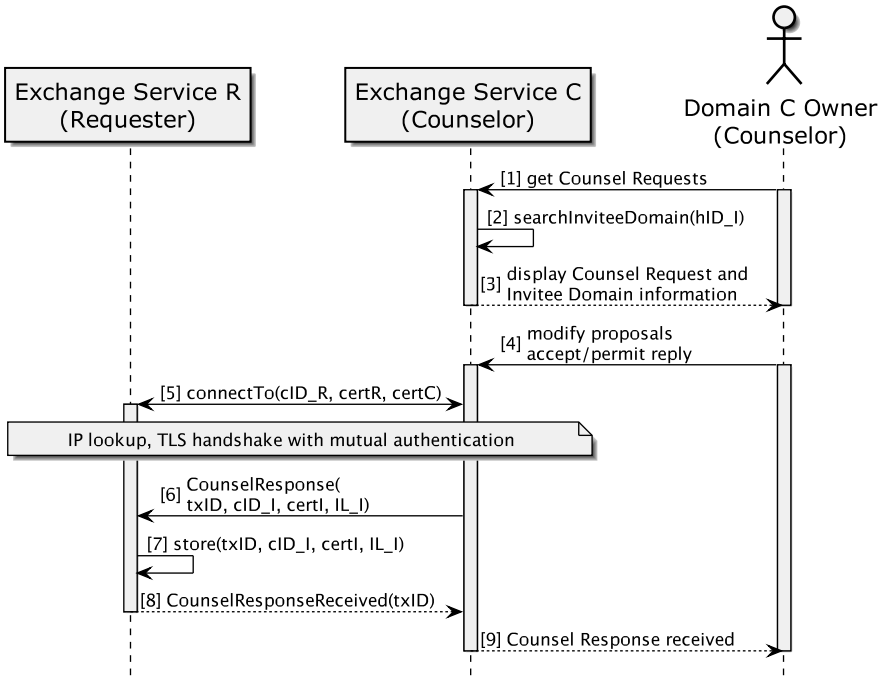


Figure 6.12: Internet Trust Exchange (details), phase 2: The Counselor provides information about eligible Domains.

Phase Three: Exchange Request and Authentication of Requester

In the third phase of the protocol, see Figure 6.13, the Requester Domain's Exchange Service contacts the Invitee Domain's Exchange Service using the cryptographic identity (cID_I) she received previously from one or several Counselors in Exchange Replies and requests the Trust Exchange.

Typically, the Counselors should propose only one Invitee Domain. But due to ambiguous human readable identities sent in the Counsel Requests (hID_I) or other problems Counselors might propose more than one Domain. To minimize the risk of establishing an erroneous Trust Relationship, the answers of the Counselors are ranked using the metric that will be introduced in the following Section 6.3.3. This ranking will help the user to select the correct Invitee Domain. Moreover, if one or several proposed Domains are ranked with a too small Identification Level, the Exchange Service will warn the Requester not to continue with the Trust Exchange.

A different problem is that the Invitee Domain's Exchange Service does not yet possess any trusted cryptographic credentials of the Requester that could be used to authenticate her. For this reason the Exchange Service will contact Counselor Domains and requests them to perform the authentication.

In the following we explain the details of the third phase of the Trust Exchange protocol, which is depicted in Figure 6.13.

After some time the Requester retrieves Counsel Replies stored by her Exchange Service (Step 1). In the following we assume for the sake of brevity that only Counsel Replies were received that concern the same Trust Exchange session.

The Exchange Service ranks the proposed Domains by their cID_I and their Identification Level (Step 2). The service then displays the results to the Requester (Step 3), who will typically select the best-ranked Domain (Step 4).

The local Exchange Service will now establish a secure connection to the Invitee Domain's Exchange Service using cID_I and the Certificate of the Invitee Domain (Step 5). Now the Exchange Request is sent that contains the Requester Domain's cryptographic ID (cID_R), the Domain certificate certR, the cryptographic identities of all involved Counselor Domains (Step 6), and the txID. The Invitee Domain's Exchange Service stores (Step 7) and acknowledges the request (Steps 8 - 9).

Now the Invitee Domain's Exchange Service tests if there is a Trust Relationship between the own Domain and the Counselor Domain or Domains named in the Exchange Request, which should be the case. The Exchange Service then establishes a secure channel to the Counselor's Exchange Service (Step 10) and sends the Authentication Request (Step 11). This request contains the cryptographic identity cID_R of the Requester Domain and txID.

The Counselor's Exchange Service searches in its database if a Trust Exchange session identified by txID exists for cID_R (Step 12). If this session exists, the legitimacy of the Exchange Request is confirmed. The Counselor Domain's Exchange Service communicates this fact by sending a positive Authentication Response to the Invitee's Exchange Service (Step 13). Furthermore, the Authentication Response contains the Identification Level of the Requester (IL_R) and txID.

The Invitee Domain's Exchange service acknowledges the reception of this message (Step 14) and stores the received information (Step 15).

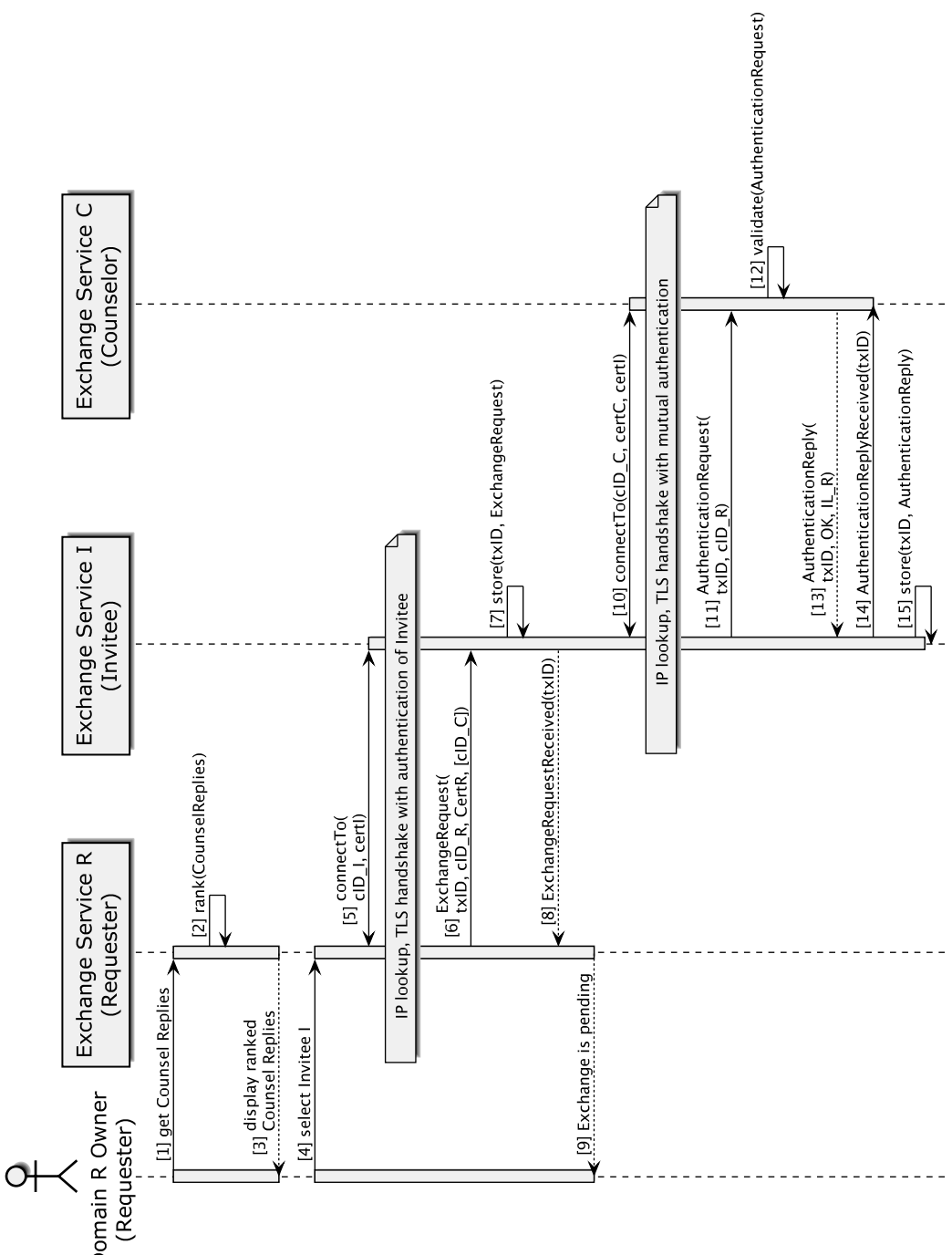


Figure 6.13: Internet Trust Exchange (details), part 3: The requester contacts the Invitee's Domain and asks for exchanging trust.

Phase four: Finalization

In the last phase of the Trust Exchange, depicted in Figure 6.14, the Invitee Domain's Exchange Service computes the Identification Level of the Trust Relationship to be established using the metric explained in Section 6.3.3 and asks its owner if the Trust Relationship should be accepted. The purpose of this step is again to assist the Domain Owner and to protect her from establishing a Trust Relationship with a Domain whose Identity is not assured. Furthermore, the Domain Owner must have the choice to accept the new Trust Relationship or not.

In the following the details of this phase are described:

Up to now, the Invitee herself was not involved into the Trust Exchange process but only her Exchange Service. In Step 1 the Invitee requests the list of pending Trust Exchanges from her Exchange Service. For brevity and simplicity we assume that only one Exchange Request is unanswered at this time.

Before displaying the pending Exchange Requests, the Invitee Domain's Exchange Service computes the IL of the newly Trust Relationship to be established (Step 2) and displays the result to the Invitee (Step 3). The displayed data also contains information about the Requester, like her human readable identifier. Based on the displayed information the Invitee decides whether to accept the Trust Exchange or not (Step 4).

Now the Invitee Domain's Exchange Service connects to the Requester Domain's Exchange Service using cID_R and establishes a secure connection, which can be mutually authenticated as the Domain Certificates are already exchanged at this point in time (Step 5). Now the Exchange Service acknowledges the Invitee's willingness to participate in the Trust Relationship (Step 6). The Requester Domain's Exchange Service stores all information belonging to the newly established Trust Relationship (Step 7) and acknowledges the receipt of the Exchange Response (Step 8). Finally the Invitee Domain's Exchange Service also stores all information related to the new Trust Relationship (Step 9) and the process is finished.

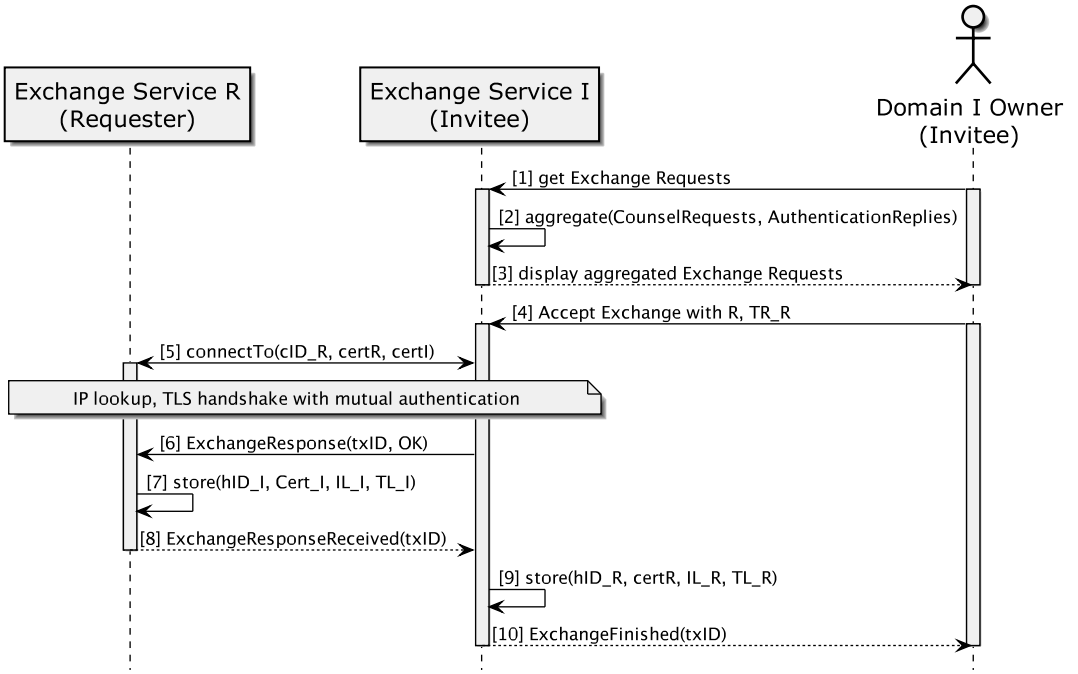


Figure 6.14: Internet Trust Exchange (details), phase 4: The Invitee accepts the Trust Exchange.

6.3.3 Trust Metric

One important requirement on the Trust Exchange mechanisms is to rate the Identification Level (IL) of the received Domain certificate, i.e., to rate the strength of the Trust Relationship. The Personal Trust Exchange utilizes a simple rating mechanism based on input given by the exchanging users. This is a valid approach as the identification of the exchange partner is implicitly (partners know each other) or explicitly (identity card) performed by the users. The same approach cannot be applied to the Internet Trust Exchange, as the exchange partners do not meet personally. Instead, the Trust Exchange is performed indirectly over a common friend (Counselor) that has already established a Trust Relationship with each exchange partner, i.e. identified each exchange partner and assigned an Identification Level to both Trust Relationships.

It would be possible to simply accept the IL assigned and propagated by the Counselor or, in case several counselors were involved in the Trust Exchange, to compute an average value and to use this average as "own" IL for the Trust Relationship with the Partner Domain. But we are convinced that this approach would be insufficient as it does not reflect over *whom* and *how* the Trust Exchange was performed. For this reason we propose the following Trust Metric.

If only one Counsel Reply[4] was received, the Identification Level IL_{RI} of a Trust Relationship between Requester (R) and Invitee (I) can be computed as follows, see Formula 6.1.

$$IL_{RI} = IL_{CI} \cdot \frac{RL_{RC}}{RL_{Max}} \cdot \frac{IL_{RC}}{IL_{Max}} \cdot d \tag{6.1}$$

Explanation of factors (also see Figure 6.15): IL_{CI} is the propagated Identification Level of the Trust Relationship between Counselor C and Invitee I. RL_{RC} and IL_{RC} are the Identification and Reputation Levels of Counselor C assigned by the Requester R in the range from 0 to 10. IL_{Max} and RL_{Max} are the maxima for the Identification and Reputation Level, i.e. the value 10. d is a dampening factor in the range from 0 to 1 to additionally decrease IL_{RI}.

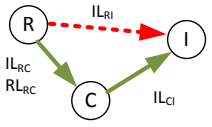


Figure 6.15: IL and RL between Domains

Interpretation of formula: The credibility of a propagated IL_{CI} depends on two properties of the Trust Relationship to the Counselor. First, the Reputation of the Counselor, i.e. how sure the Requester can be that the Counselor assigned the right IL to the propagated Trust Relationship. Second, the IL the Requester has assigned to the Counselor by himself. So the formula expresses that the IL_{CI} of a Trust Relationship established over a *securely* identified Counselor that has a *very good* reputation has *high* credibility. If the Counselor is rated *not* to be very reliable or was *insecurely* identified, IL_{CI} is decreased by the formula. Additionally, the dampening factor d decreases the ILs established over the Internet as we argue that exchanging trust indirectly can never be as reliable as exchanging trust directly in person. This is, for instance, because the Counselors might propose the wrong invitee due to an ambiguous Counsel Request. Hence, the maximal IL_{RI} of a Trust Relationship that can be established over the Internet is $IL_{Max} \cdot d$. In the current implementation we use the dampening factor $0,9$ so the maximum IL will be $10 \cdot 0,9 = 9$.

[4]Please note that the same calculation can be applied to rate IL_{IR} based on one Authentication Reply

If more than one Counselor has sent Counsel Replies[5], the single ILs can be aggregated as follows, see Formula 6.2:

$$IL_{RI} = \frac{\sum_{C \in Counselors} IL_{CI} \cdot \frac{RL_{RC}}{RL_{Max}} \cdot \frac{IL_{RC}}{IL_{Max}}}{count(Counselors)} \cdot d \tag{6.2}$$

The formula computes the algebraic average of the IL_{CI} values propagated by each single Counselor. As in Formula 6.1 each IL_{CI} is weighted by the Counselor's Reputation and Identification Level. The dampening factor d could be set to 0,9 as before but we propose to adjust it dynamically according to the number of Counselors involved in the Trust Exchange. If more than three Counselors were involved, the dampening can be set to 1 (resulting in no dampening at all). This is reasonable as the risk to establish a Trust Relationship to an incorrect Domain is strongly reduced if multiple Counselors proposed the same Domain.

In Figure 6.16 we have depicted a simple example scenario with IL and RL values where R has established a Trust Relationship to I.

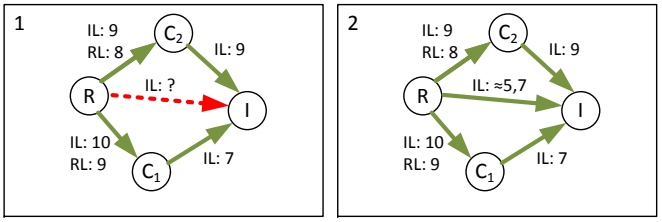


Figure 6.16: Example: Aggregation of propagated Identification Levels.

The IL_{RI} of the new Trust Relationship can be calculated by inserting known IL and RL values and propagated IL values into Formula 6.2. d is set to 0,9 as only two Counselors participated:

$$IL_{RI} = \frac{7 \cdot \frac{9}{10} \cdot \frac{10}{10} + 9 \cdot \frac{8}{10} \cdot \frac{9}{10}}{2} \cdot 0,9 \approx 5,7 \tag{6.3}$$

6.3.4 Discussion and Evaluation

This section discusses and evaluates the Trust Exchange mechanism over the Internet. We examine if all requirements are met and additionally discuss further properties of the proposed mechanism.

6.3.4.1 Fulfillment of Requirements

Attacks on Identification/Authentication of Trust Exchange Partners (R1)

The aim of an attacker is to trick a Domain Owner (victim) into establishing a Trust Relationship with her, which might finally lead to the attacker's ability to access (abuse) resources provided in the victim's Domain. As the victim will only participate in a Trust Relationship with a friend, the attacker must find a way to impersonate such a friend. For achieving this, the attacker has two possibilities, namely she might act as malicious Requester or malicious Counselor.

[5]Please note that the same calculation can be applied to rate IL_{IR} based on several Authentication Replies

Impersonation Attacks by a Malicious Requester: A malicious Requester R_m starts the first class of impersonation attacks. In this case R_m tries to impersonate a friend R of an Invitee I and to trick I into a Trust Relationship with the impersonated R.

The first difficulty R_m has is to find a Counselor C that shares a Trust Relationship with I. As our mechanism is privacy preserving, which means that no central server can be queried that might reveal existing Trust Relationships (R5), it is difficult for R_m to find a suitable C in a targeted manner. But if R_m knows the social graph of I well, she might simply guess a suitable Counselor C for I. R_m's next problem is to establish the secure connection to the Exchange Service of the guessed C. This is because R_m does neither know C's cryptographic identity cID_C, which is needed to determine the IP of C's Exchange Service, nor does R_m share a Trust Relationship with C. For this reason R_m is not able to authenticate towards the Counselor's Exchange Service and subsequently send a Counsel Request. Finally the attempt to attack I fails (R3).

R_m could also try to directly send an Exchange Request to I claiming she is R without first contacting any Counselor. Without a Counsel Reply for I sent from a Counselor, R_m is missing the cryptographic identity of the attacked I. Again, the missing information will complicate the attack. But R_m could have obtained I's certificate by any other means, then compute cID_I by herself and obtain the IP of I's Exchange Service. I's Exchange Service cannot differentiate between legitimate or illegitimate Exchange Requests. For this reason the service will process R_m's Exchange Request, meaning the Exchange Service will contact all Counselors named in the Exchange Request and send Authentication Requests to all of them. R_m will fail the subsequent verification of the Exchange Request she sent to I, as R_m is either unable to name suitable Counselors or, in case she guessed one or several Counselor correctly, the Counselors will not authenticate the Exchange Requests. R_m's attempt to establish a Trust Relationship to I will fail again.

Impersonation Attacks by a Malicious Counselor: The second class of impersonation attacks might be started from a malicious Counselor C_m. But acting as Counselor is only possible when other Domains have already established a Trust Relationship to a Domain before. The highest likelihood that a Counselor becomes malicious emerges from the threat that an attacker impersonates a formerly "honest" Domain. In this case, the attacker is able to completely control and abuse existing Trust Relationships of this Domain to other Domains. All these Domains, either acting as Requester or Invitee that trust the compromised Domain to be a honest Counselor, are subsequently endangered.

The first option the attacker has, see Figure 6.17(a), is to wait for incoming Counsel Requests for an arbitrary Invitee I sent by a Requester R. The attacker is able to impersonate R's friend I as he is able to claim in a Counsel Reply that a Domain, which is also controlled by the attacker, is owned by the searched I.

Vice versa, if the attacker does not want to wait for Counsel Requests, she can act additionally as malicious Requester herself, see Figure 6.17(b). She first sends an Exchange Request to some Invitee I claiming to be I's friend R. Subsequently, the Invitee will send an Authentication Request to verify R's identity to the malicious Counselor who will happily confirm the impersonated identity.

Attacks that involve a malicious Counselor are the strongest attacks on the Internet Trust Exchange protocol as the Counselor is expected to be a trusted party. Therefore this kind of attack cannot be prevented by careful protocol design but only mitigated when multiple counselors are used and the strength of the proposed Trust Relationship is rated by the Trust Metric.

(a) Only the Counselor is malicious. R is tricked into a Trust Relationship with a faked I (I_m)

(b) Requester and Counselor are malicious. I is tricked into a Trust Relationship with a faked R (R_m)

Figure 6.17: Attacks on the Internet Trust Exchange by a malicious Counselor.

Meaningfulness of the Identification Level Rating (R2)

As explained before, we use a Trust Metric during the Internet Trust Exchange to compute the Identification Level (IL) of a proposed Domain based on IL ratings propagated by Counselors. The result of this calculation is used to rank different proposed Domains, to warn a Domain Owner in case insufficiently identified Domains were proposed, and finally as IL value of the new Trust Relationship.

Each Trust Relationship, also those established via the Internet Trust Exchange, can be propagated when the Domain itself acts as Counselor. The IL of the Trust Relationship will also be propagated and reused by other Domains.

We argue that the proposed Trust Metric is well suited for our scenario as it reflects various properties of the involved Counselors, namely their own IL and RL, and the threat imposed by accidentally or willingly given wrong Domain proposals by a small amount of Counselors.

A Domain proposal given by a Counselor can only be meaningful, if the Counselor herself was identified without doubt before. Otherwise the threat rises, that a Counselor accidentally or willingly gives wrong Domain proposals. Therefore, the Trust Metric weighs the IL propagated by the Counselor's IL. This finally results in a reduction of a propagated IL if the Counselor's IL is low.

A Domain proposal can also be meaningful only, if the Counselor is reliable, which means expected to check identities of other Domain Owners during a Personal Trust Exchange well and also assign the correct IL to a new Partner Domain. Therefore, the Trust Metric also weighs the IL propagated by the Counselor's RL. This finally results in a reduction of a propagated IL if the Counselor's RL is low.

The more Counselors proposed the *same* Domain, the lower is the threat that the proposed Domain is incorrect. For this reason, the Trust Metric takes the amount of Counselors that have proposed the same Domain into account. For this purpose, the Trust Metric decreases final ILs only if one or two Counselors proposed a Domain. If more than three Counselors proposed the same Domain we argue that it is not necessary anymore to decrease the final IL.

The Trust Metric works best if many Counsel Replies are available. But possibly not many counselors exist at all or not many Counselors have replied to a Counsel Request at the point in time the Requester decides to continue with the Trust Exchange and send out Exchange Requests to proposed Invitee Domains. For this reason, the Internet Trust Exchange is no process that should be performed in a hurry. Such Trust Exchanges might be less reliable as those who ran longer and gave more time to multiple Counselors to send a Counsel Reply. Nevertheless, the Trust Exchange mechanism might still uses Counsel

Replies received after the Trust Exchange is already finished to update the IL of the Partner Domain. If a serious problem was encountered, the Domain Owner is warned and the already accepted Trust Relationship is revoked. For instance, if a Counselor proposed a different Domain than other Counselors, this is strong evidence that there is a problem.

The Trust Metric is also extensible and highly customizable, which allows users to adapt the rating according to their wishes. Domain Owners that have high demands on security might, for instance, lower the dampening factor d (resulting in a higher dampening) and chose to only fully trust a Domain proposal if, for instance, five or more Counselors proposed the same Domain.

Further Attacks

Besides impersonation attacks, other options exist how the Trust Exchange might be attacked and abused.

Flooding an Exchange Server (DoS): A possible option an attacker has is to perform a denial of service attack against the Exchange Services of Domains by flooding them with Exchange Requests or Responses. This attack becomes possible, as the Exchange Service is publicly reachable. Nevertheless, it is difficult for an attacker to perform a targeted attack on a specific Domain's Exchange Service, as she does most likely not know the cryptographic identity and possess the certificate of the Domain.

By performing a DoS attack on the Exchange Service, the attacker does not gain much except disturbing the service. The Domain Owners themselves are never affected, which means they are never queried for permission, etc. (R8). The reason for this is that the Exchange Service will immediately deny all requests or responses that cannot be authenticated. The only request an attacker can send to an Exchange Service is the Exchange Request. This request cannot be authenticated by the own Exchange Service but requires sending an Authentication Request to a Counselor. Again, the request will be discarded silently when the Counselor was not able to authenticate the request without further disturbing the Domain Owner.

By performing a DoS attack on honest Counselors, a malicious Counselor might increase her chances that an impersonation attack (see Figure 6.17) succeeds. The attacker might flood honest Counselors with arbitrary requests, which might shut down the attacked Exchange Service. The effect of this attack might be that an Invitee Domain that wants to send an Authentication Request to her Counselors will only obtain an Authentication Response sent by the malicious Counselor. The threat that the impersonation succeeds is increased.

Man-in-the-Middle (R3): A Man-in-the-Middle might try to interfere with the Internet Trust Exchange protocol and, for instance, insert bogus information (cryptographic identity, certificate, identification level) in protocol messages. This would be a dangerous attack especially if responses sent by Counselors could be modified, as the Domains trust these replies and establish the new Trust Relationship based on the information contained in these responses.

Luckily, all network connections between Requester and Counselor and between Counselor and Invitee are secured by TLS/SSL using the already exchanged certificates. The last steps of the Internet Trust Exchange protocol, which run between Requester and Invitee, are also already secured using the certificates exchanged via the Counselor (R3).

User-Centrism (R4) and Attacks on Privacy (R5)

The Internet Trust Exchange protocol meets the requirement on user-centrism as no "classic" trusted third parties (public CAs/PKI, GPG/PGP key server) are needed. Additionally no central entity is needed that stores privacy relevant information, such as certificates on GPG/PGP key servers that might reveal (social) relationships between Domains. Also the privacy of Trust Relationships of a Counselor's Domain to other Domains is protected during the Trust Exchange. The Counselor needs to explicitly agree to take part in the Trust Exchange, as this will reveal the existence of a social relationship between herself and the Invitee.

Security and Privacy vs. User-Friendliness

The requirements we defined on user-friendliness partially conflict. The Trust Exchange mechanism could be designed more responsively (R9), with a higher degree of automation (R6), and with less interaction with the Domain Owner (R8) if we would waive requesting the permission to take part in the Trust Exchange. In other words: user-friendliness and security/privacy conflict.

Nevertheless we are convinced that the solution found is a good middle course. The owner of a Domain keeps full control on what is happening (R7) but does not need to answer too many inquiries by the system (R8). The only inquiries the Domain Owner needs to answer control which Trust Relationships are propagated to other Domains and from which Domains she accepts Exchange Requests. The former inquiry protects, as previously discussed, the privacy of existing Trust Relationships (R5). The latter is especially important to prevent that Trust Relationships are established to undesired Partner Domains.

Finally, the Internet Trust Exchange is flexible and can be used between all Domain types, also Domains, which are unrelated to home networking (R10).

6.3.4.2 Deep Search in the Trust Graph

The requirement to obtain the privacy of the Domain (R5) and to obtain the Domain Owner's consent (R7) make it difficult to implement an automated mechanism that would "flood" Counsel Requests over the trust graph, see Figure 6.18. On every hop the consent of the Domain Owner would be needed, which takes some time to obtain. But by using such an approach it would be possible to query also those Counselors, which do not have a direct Trust Relationship with the Requester herself, i.e., the likelihood that the Invitee can be reached would be increased.

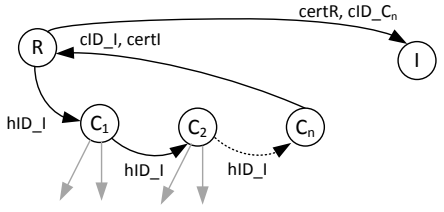


Figure 6.18: Flooding of Counsel Requests over the trust graph.

On first sight this seems to be a severe limitation as Trust Exchanges can only be performed over Counselors that are one "hop" away. But flooding Counsel Requests over many hops would decrease the reliability of the established Trust Relationships quite quickly. For this reason, the question needs to be answered if Trust Relationships established over Counselors that are some hops away are useful at all.

6.3.4.3 Benefits and Drawbacks

The Internet Trust Exchange complements the Personal Trust Exchange as it implements the missing ability to establish Trust between Domains without a personal meeting. The gain in flexibility comes with a slight loss of security, as third parties are involved in the establishment of the Trust Relationship.

For this reason, Trust Relationships established over the Internet are not as strong as Trust Relationships established personally. This fact is reflected by the Trust Metric, which automatically assigns an appropriate IL to the established Trust Relationship.

6.3.4.4 Applicability

The contributions of this chapter are not limited to the home networking context. Also in companies, larger building networks, etc. the presented approach and mechanisms can be used to establish trust between Domains. A possible scenario might be departments of different companies that need to collaborate and share resources for a certain project.

6.3.4.5 Open Issues and Future Work

Neither IL nor RL reflect the likelihood that a Domain's Exchange Service gets compromised by an attacker and turns malicious. To reflect such a threat, a reliable and measurable property of an Exchange Service or a Partner Domain must be found first. We discuss the usage of Trusted Computing technology in Sections 12 and 13 to protect the integrity of a Domain and to protect a Domain CA's private key against theft. These mechanisms render various attacks more difficult and their usage is provable. The Trust Metric might be extended by an additional factor that dampens the IL of a proposed Domain if this Domain does not use the mentioned protection mechanisms. Vice versa, Domains that use these technologies might be preferred, as the likelihood becomes lower that these Domains are compromised. Up to now, we do not use this information in the metric. Elaborating this idea might be part of future work.

6.4 Conclusion

An important issue that was not solved so far is how Domains that do not belong to the same home network can authenticate each other's entities. This inability is caused as there is no certificate chain (trust path) between such Domains. We argued that classic PKIs based on the X.509 or PGP/GPG trust model are not applicable to the home scenario, have issues concerning trustworthiness (X.509) and privacy (PGP/GPG) and should therefore not be used to create the missing trust path between Domains.

For this reason we introduced in this chapter the basic approach and two different implementations of a mechanism called Trust Exchange. Exchanging Trust between Domains will allow the authentication of entities between Domains as the Partner Domain's certificates are first exchanged and then distributed to local entities. The entities that possess the exchanged certificate are now able to authenticate entities that belong to the Partner Domain. This ability is highly important, as it is required to share services across Domains.

The first Trust Exchange mechanism, called Personal Trust Exchange, implements the secure and effortless exchange of Domain certificates between two members of Partner Domains at a personal meeting. As we have discussed, this approach is the most secure and reliable way in which two Domains may establish trust with each other. Nevertheless we see the problem that trust can only be established when the Partner Domain members are able to meet personally, which is not always possible.

For this reason, we proposed a second Trust Exchange mechanism, called Internet Trust Exchange that is based on the idea to establish new Trust Relationships to Domains using already existing Trust Relationships to other Domains. These so-called Counselor Domains act as a specific type of trusted third party during the Internet Trust Exchange and help exchanging certificates between Partner Domains. As the result of the Internet Trust Exchange depends on the credibility of Counselor Domains, the Internet Trust Exchange cannot be as secure and reliable as the Personal Trust Exchange. This fact is reflected by a Trust Metric that rates the quality of Trust Relationships established over the Internet and also warns Domain Owners in case a desired Trust Relationship cannot be established in a sufficiently reliable manner.

No matter how high the quality of a Trust Relationship established over the Internet was rated by the metric, it should always be strengthened when there is a good opportunity to do so, e.g., when members of the Partner Domains meet. For this purpose the Personal Trust Exchange Mechanism can be used.

Both Trust Exchange mechanisms were built around the fundamental requirements to be secure, to respect the privacy of users, and to be simple to use. As we have discussed above, we were able to fulfill these requirements with both mechanisms.

As this chapter delivered the final component for an identification/authentication scheme for unmanaged networks, we can summarize its properties and compare this "hybrid" identification/authentication scheme's applicability to classic Trust Models/PKIs regarding the envisaged application scenario, see Table 6.1.

Key Findings and Contributions

- ▷ The establishment of a **Trust Relationship** between Domains that belong to different homes may not involve a public PKI (X.509 PKI or PGP/GPG Web-of-Trust) as various drawbacks (trust, privacy) would result.
- ▷ A better approach how trust between Domains can be established is strictly user-centered, which we call **Trust Exchange**.

C6.1 The **Personal Trust Exchange** mechanism depends on no third parties as it automates the secure exchange of Domain CA certificates between Domain members during a personal meeting. This method offers the most secure mutual identification and certificate exchange, but cannot be performed when no personal meeting is possible.

C6.2 The **Internet Trust Exchange** mechanism is an alternative technology that exchanges Domain certificates semi-automated and in a secure and privacy preserving manner over the Internet by leveraging already existing indirect Trust Relationships between Domains.

C6.3 The **Trust Metric** evaluates the strength of a Trust Relationship during the Internet Trust Exchange and prevents that a Trust Relationship is established to an insufficiently identified/authenticated Domain.

	Public X.509 PKI	PGP/GPG WoT	**Hybrid Trust Model (This thesis)**
Trustworthiness	✗ Depends on externally controlled, possibly compromised/treacherous CAs. Hence, trustworthiness is impossible to rate for users.	❍ User centric, but only highly trustworthy when used correctly.	✓ Highly trustworthy due to user centrism and automated/guided assistance software that excludes user errors.
Privacy preserving	❍ CAs know owners of certified keys, but no information published.	✗ User centric, but severe privacy issues due to published keys and certificates.	✓ User centric, no information public, user need to explicitly permit privacy critical actions.
Usability/ User-friendliness	❍ User-friendly if certificates issued by a public CA are consumed (web browser). Difficult when certificates need to be issued by user controlled CAs.	✗ Difficult to understand concept, difficult to use software.	✓ Complexity entirely hidden, software components take care for difficult to understand or perform steps.
Applicability to the home scenario	❍ Flat, does not reflect network structure well, widely used certificate format.	❍ Hierarchical identifiers possible, certificate format mostly used for mail communication.	✓ Hierarchical identifiers, widely used certificate format.

Table 6.1: Comparison of applicability of classic Trust Models vs. the Hybrid Trust Model of this Thesis regarding the envisaged application scenario.

7. Authorization

In the previous chapters various components of the Guided Security Management System were presented that enable identification and authentication in unmanaged networks, such as home networks. Chapter 4 introduced the design of an identification/authentication scheme suitable for Domains. A mechanism that enables a Domain Owner to effortlessly and securely register entities that belong to her Domain, which means to generate and distribute asymmetric keys certified by the Domain's CA, was designed in Chapter 5. Finally, mechanisms were presented in Chapter 6 that allow Domain Owners to establish Trust Relationships to Partner Domains that do not belong to the own home network.

The ability to identify and authenticate entities within a Domain and across *friendly* Domains is already a great improvement for the security of unmanaged networks. Friendly Domains are either User Domains of the same home or Partner Domains, which can either be User or Home Domains. However, the question must be answered if access control based on authentication only is already sufficient.

It appears quite reasonable that an entity is allowed to access all services that belong to its *own* Domain. This is due to the fact that all entities of a Domain have the same owner, e.g., the home owner or a certain resident. In this case, an additional authorization step for access control is usually not necessary.

Unlike the first case, it becomes highly questionable if all services of a Domain may be accessed by an entity that belongs to *another*, but friendly, Domain, especially if this Domain does not even belong to the same home. In such cases an additional authorization step for access control is urgently needed.

Another limitation of access control based on authentication only is the lack of flexibility and its coarse granularity. For instance, once an entity was successfully authenticated by a service, it would be allowed to use *all* functionalities provided by this service (e.g. set the room temperature, disarm the alarm system,...) or to access *all* resources this service offers (e.g. all media files, ...) when no subsequent authorization is done.

This brief explanation already shows that access control, which solely uses authentication of entities accessing a service is insufficient. In order to avoid possible security problems caused by the just described limitations the present chapter investigates how an authorization component that adds the ability to control access in fine granularity and with high flexibility can be integrated into the Guided Security Management System.

The focus of this chapter, authorization, has already been defined as Requirement RB "Flexible and Fine Grained Authorization" in Section 2.4 and is one of the two overall functional requirements on the envisaged access control system. Requirement RE "User Friendliness and Security per Default" again influences the solution to this problem.

Chapter Structure

The present chapter is structured as follows: Section 7.1 defines requirements on the authorization component. In Section 7.2 we outline our approach how the XACML authorization framework can be integrated into the Guided Security Management System and how a Domain Owner can specify access control settings with ease. Section 7.3 details the most important aspects of XACML integration. Design patterns for XACML policies suitable for Domains are discussed in Section 7.4. A solution that allows inexperienced users to create and manipulate XACML policies is detailed in Section 7.5. The findings of this chapter are discussed and evaluated in Section 7.7. The chapter is finally concluded in Section 7.8.

Acknowledgments

This chapter contains results developed during the internship of Sunil Kumar Ghai[1] and in the Guided Research Project of Paulo Henrique Azevêdo Filho [45]. These works were assigned and supervised by the Author in the context of his doctorate.

7.1 Requirements

Based on the analysis of properties of home networks, see Section 2.2, the following requirements on the authorization component of the Guided Security Management System are defined:

- **RB.1: Efficient Domain-Wise Access Control:** Typically it does not matter which device of a Domain is used to access a service as long as the Domain belongs to a person that has the right to access the service. For this reason access control must be performed Domain-wise as a default.
- **RB.2: Entity-Wise Access Control:** A deviation from the default might be needed when a Domain Owner desires to grant access rights to an *individual* device. For instance, rights to access a security critical service might be granted only to a device that has special hardware/software features, which make it difficult to compromise the device. For this reason, access control must also consider individual entities.
- **RB.3: Fine-grained Access Control:** Access rights need to be expressible in fine granularity. It must be possible to assign the right to access
  - an individual service (e.g. an A/V streaming service or the home control service),
  - an individual resource (e.g. a certain media file),
  - an individual action (e.g. set the room temperature, disarm the alarm system)

  to a Domain or entity.

[1] Associated to the Delhi College of Engineering and intern at the Technische Universität München.

- **RB.4: Compatibility to the Identification/Authentication Scheme:** The authorization mechanism must leverage the Identification/Authentication Scheme presented in this thesis.
- **RB.5: Centralization:** In a Domain a central point must exist where access control settings are created, modified and stored.
- **RB.6: User-Friendly and Fault-Resistant Access Control Management:** The management of access control settings must be done in a user-friendly and also fault-resistant manner. For this purpose, complicated technical details need to be hidden from the Domain Owner resp. the Domain Owner needs to be guided through the creation and modification process of access control settings.

7.2 Approach and Basic Architecture

XACML is a standardized authorization framework that offers a highly expressive and flexible authorization policy language on one hand and the architecture of authorization services on the other hand, see Section 3.4.3. Our approach to add fine-grained authorization to unmanaged networks is based on the integration of XACML technology into the Guided Security Management System of Domains.

The first difficulty that has to be solved is the integration of XACML authorization into existing "legacy" services and to connect this technology with the identification/authentication scheme. The second difficulty is to enable the Domain Owner to express her wishes how access should be controlled in her domain as a XACML Policy Set.

Figure 7.1 depicts the components of the Guided Security Management System and XACML related components that exist in every Domain.

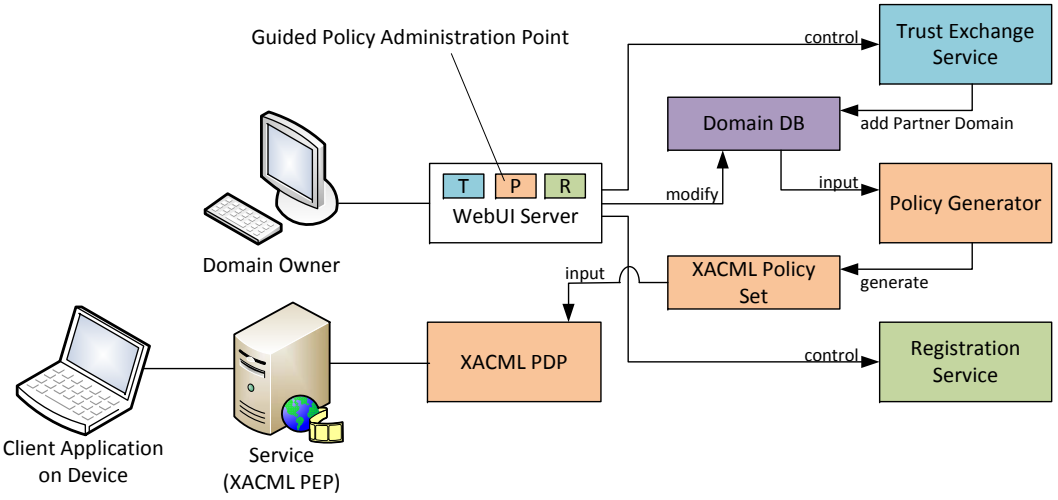


Figure 7.1: Architectural Overview: Components of the Guided Security Management System and XACML services in a Domain.

The first problem can be solved using a technology we called *TLS Handshake Interception*. TLS Handshake Interception basically stops (intercepts) a TLS handshake just after the authentication of the accessing entity and computes the cryptographic identity of the accessing entity. This identity is then included into a XACML request and sent to the *Policy Decision Point (PDP)* of the Domain. The PDP decides whether to grant access or not based on its *Policy Set*, communicates the result to the Service, which finally enforces the PDP's decision. For enforcing the PDP's decision the TLS Handshake Interception manipulates the last step of the TLS handshake, see Section 7.3.

The syntax of the XACML policy language is complicated and confusing. For this reason it is almost impossible that inexperienced users implement a useful concept how access should be controlled in their domain in a semantically and/or syntactically valid XACML Policy Set. Hence, we designed and implemented a *Guided Policy Administration Point*, which is a service that semi-automates the generation of XACML Policy Sets and is especially adapted to the needs of Domains. The Domain Owner interacts with this service as she is used to, i.e., via a Web-based GUI (WebUI). The WebUI displays the contents of the Domain DB, namely known friendly Domains and available services, and allows the Domain Owner to simply "connect" Domains with those services that may be accessed. Based on this knowledge and the Domain Owner's settings, the *Policy Generator* produces a XACML Policy Set, for details see Section 7.5.

7.3 TLS Handshake Interception

In order to integrate XACML-based authorization into newly created or already existing services and to enable XACML to use the cryptographic identities of devices and services we developed *TLS Handshake Interception*.

The basic idea of Handshake Interception is to stop ("intercept") the TLS four-way handshake at the side of the service (XACML PEP) just after the client certificate was successfully verified. Instead of finishing the TLS handshake, the cryptographic identity (cID) of the client is computed based on the chain of certificates it presented. The cID is then used as Subject in an XACML request.

7.3.1 Basic TLS Handshake Interception

Figure 7.2 details the basic variant of the TLS Handshake Interception. Only a small portion of code needs to be added to the implementation of legacy services. This extra code is needed to compute the cryptographic identity of the accessing client, to connect to the *PEP Proxy* and finally to enforce the PDP's access decision. The PEP Proxy is a simple service that acts as intermediate between PEP and PDP. Its purpose is to encapsulate some XACML-related functions and to enable feedback-based learning of access rights. This mechanism is detailed in Section 7.6. As the PDP, the PEP Proxy is a central service that exists within each Domain.

In Step 1 the client starts to establish a secure connection to the service using TLS. Steps 2 - 3 depict the mutual exchange of certificates and their authentication. If the client device presents a certificate chain that can be authenticated using trusted Domain CA certificates and is able to prove the possession of the corresponding private key the authentication will succeed. The device is then regarded to be a valid member of the own Domain resp. of a friendly Domain. Now its attempt to access the service needs to be authorized.

In Step 4 the Handshake Interception pauses the TLS handshake just before the last, forth TLS handshake message is sent to the client. The public keys contained in the client's certificate chain are extracted and the cryptographic identity of the device is computed as described in Section 4.3. The cryptographic identity of the device is later used as Subject in the XACML request. For this reason we refer to the cID of an accessing device as the *SubjectID*. Now SubjectID and *ResourceID*, which is the cryptographic identity of the service itself, are sent in a simple XML-based message format to the PEP Proxy (Step 5).

The PEP Proxy creates an XACML request from the received information and sends this request to the PDP (Steps 6 - 7). The PDP evaluates the request in Step 8 and responds with a decision (Step 9). The PEP Proxy translates the decision into the XML-based massage format and sends this message to the service (Step 10).

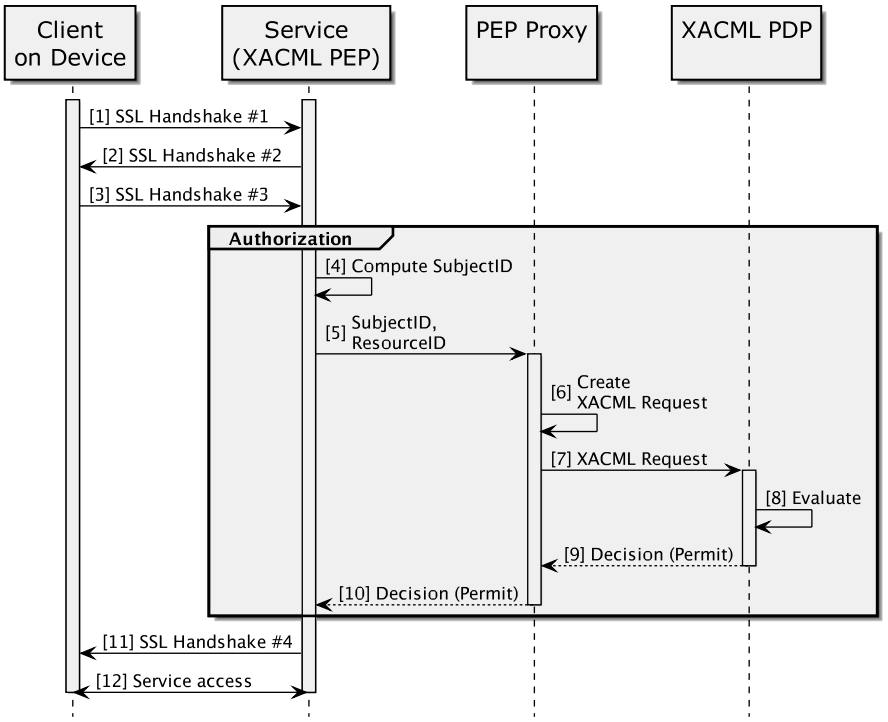


Figure 7.2: Basic TLS Handshake Interception

If the PDP decided to permit the request, the TLS handshake is continued, the secure TLS channel is established and the client may access the Service. If the request was denied, the TLS handshake is aborted using the TLS alert `access_denied`.

7.3.2 Extended TLS Handshake Interception

The basic TLS Handshake Interception will authorize the client for the whole lifetime of a TLS session. This means that it is not possible to authorize requests in fine granularity, for instance, requests to a specific resource or an action that might be triggered. For some services this might be sufficient already, for services that require fine-grained authorization an extended variant of the TLS Handshake Interception is proposed, see Figure 7.3.

In contrast to the basic variant the integration of this technique into a legacy service is more complicated as modifications need to be made at different places in the implementation. Code added to an implementation is partially service dependent, for instance, a function that extracts information from service specific protocol messages.

As before, the client starts to establish a secure TLS channel to the service. The TLS handshake is intercepted in Step 3 and the SubjectID of the accessing client is computed (Step 4). In contrast to the simple variant of the protocol, the TLS handshake is immediately completed and the secure TLS channel established (Step 5).

Now, the client sends a request over the secure channel to the service (Step 6). For instance, when a web service is accessed a `HTTP GET` for a particular web site (`foo.php`) that triggers a particular Action (`action=baz`) will be sent. Information is now extracted from the request, in the example, the requested web site (`foo.php`) and the *Action* (`baz`). The SubjectID, ResourceID and Action are added to the XML-based message, which is then sent to the PEP Proxy. Steps 9 - 13 will authorize the request as explained before.

In case the PDP permitted the request, the service will process the request (Step 14) and send the result to the client (Step 15). If the request is denied, a suitable response is sent

that aborts the process. In the example of a web service, a HTTP message with the return code `401 (Unauthorized)` would be appropriate.

Steps 6 - 15 are repeated in order to authorize every request sent by the client.

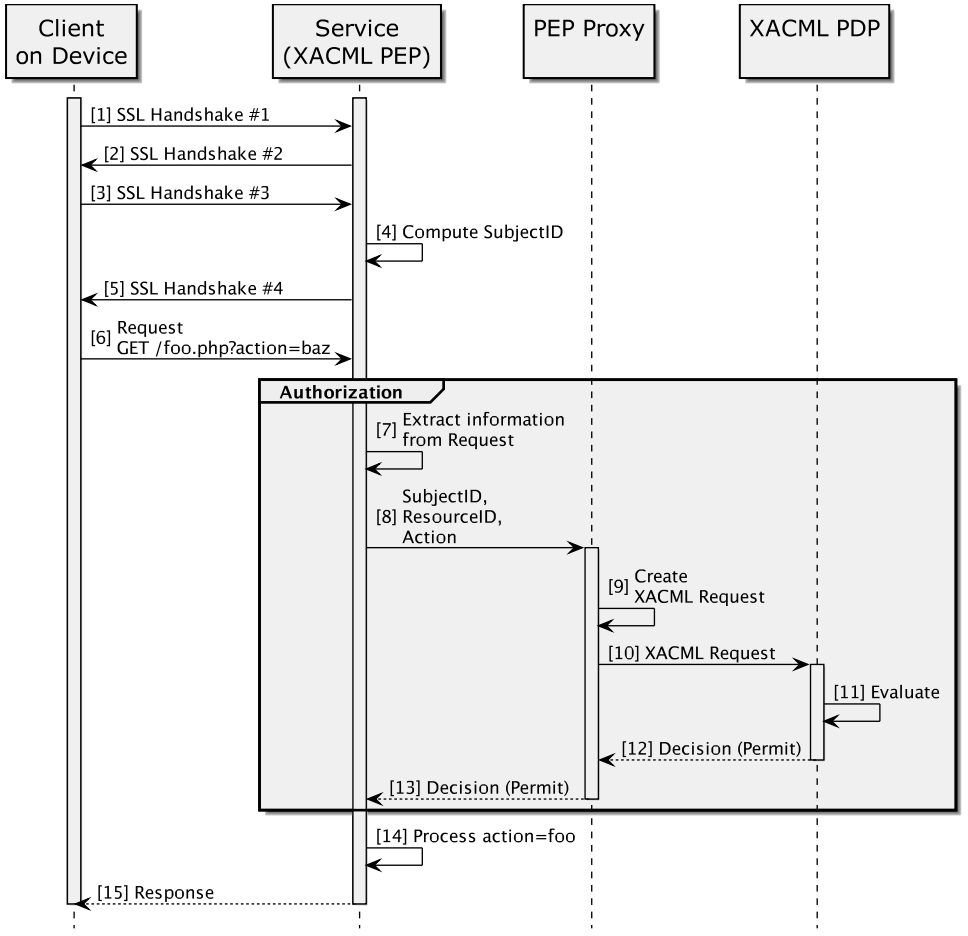


Figure 7.3: Extended TLS Handshake Interception with subsequent fine-grained authorization

7.4 Design Patterns for XACML Policy Sets for Domains

As explained above, TLS Handshake Interception extracts public keys from the certificate chain presented by an accessing entity and computes the entities cryptographic identity based on this data. The computed cryptographic identity is then used as SubjectID in a XACML request, which is then matched by the PDP to a specific XACML Policy in the Domain's Policy Set based on the Subjects of all Policies. In this section the design of Policies suitable for the Policy Set of a Domain is discussed.

Basic Considerations

As explained before, typically all entities that belong to a Domain should obtain the *same* access rights. With a flat identifier, this could be achieved by creating *many* similar Policies that all assign the same access rights to each individual entity of a Domain. But the cryptographic identifier scheme designed in this work has a hierarchical format (`HomeDomain.UserDomain.Entity`). For this reason it is possible to create a *single* Policy that assigns access rights to a Domain, i.e., to all entities that belong to this Domain at once.

In order to implement such Policies a way must be found how this hierarchic structure can be leveraged, i.e., how Policies can be created that match to a specific part of the SubjectID contained in a XACML request. For instance, to `HomeDomain.UserDomain.*` or `*.UserDomain.*`, whereby "`*`" is a wildcard operator. For this purpose the XACML string matching function `string-regexp-match` can be used, which matches Policies using regular expressions.

By means of the hierarchical identifier of an accessing entity, six different cases can be differentiated, that need to be reflected by different Policy design patterns, see Table 7.1.

Table 7.1: Differentiation of accessing entities by means of their identity.

Case	HomeDomain	UserDomain	Entity is Member of
1	local	own	own User Domain
2	local	any	any User Domain of local Home Domain
3	local	-	local Home Domain
4	any	trusted	Partner User Domain
5	trusted	any	any User Domain of Partner Home Domain
6	trusted	-	Partner Home Domain

Design Pattern for an XACML Policy for own Entities

Entities that belong to the own User Domain have an identity in the form `localHomeDomain.ownUserDomain.*` (Table 7.1, Case 1). A Policy that matches identities of this form can be created by setting the Policies Subject to `localHomeDomain.ownUserDomain.[0-9a-zA-Z]*`.

In this particular case, the Policy must fulfill another property. The Policy must grant access to all services that belong to the own User Domain. This can be implemented by setting the ResourceID of this Policies Rule to `localHomeDomain.ownUserDomain.[0-9a-zA-Z]*`.

The design pattern of this Policy type is depicted in Listing 7.1.

```
1  <Policyset>
2   <Policy>
3    <Target>
4     <Subject>localHomeDomain.ownUserDomainID.[0-9a-zA-Z]*</Subject>
5    </Target>
6    <Rule permit>
7     <Target>
8      <Resource>localHomeDomain.ownUserDomainID.[0-9a-zA-Z]*</Resource>
9     </Target>
10   </Rule>
11  </Policy>
12 </Policyset>
```

Listing 7.1: Design pattern for a XACML Policy for own entities

Design Patterns for XACML Policies for Entities of the own Home

In order to control access of Entities that belong to any User Domain of the local home (Case 2 in Table 7.1) a Policy with the Subject `localHomeDomain.[0-9a-zA-Z]*.[0-9a-zA-Z]*.` is needed, as identities of such entities have the form of `localHomeDomain.*.*`

In this case, access may not be granted to all services in the own Domain but only to a selected subset. For this reason, the ResourceIDs of specific services need to be specified in the Target of this Policies Rule, see Listing 7.2.

```
<Policyset>
 <Policy>
  <Target>
   <Subject>localHomeDomain.[0-9a-zA-Z]*.[0-9a-zA-Z]*</Subject>
  </Target>
  <Rule permit>
   <Target>
    <Resources>
     <Resource>localHomeDomain.ownUserDomain.Service_1</Resource>
     [...]
     <Resource>localHomeDomain.ownUserDomain.Service_n</Resource>
    </Resources>
   </Target>
  </Rule>
 </Policy>
</Policyset>
```

Listing 7.2: Design pattern for a XACML Policy for entities of any local User Domain

Policies for public entities of the local Home Domain (Case 3 in Table 7.1) require a Policy with a Subject set to `localHomeDomain.[0-9a-zA-Z]*`. Again, access must be granted to specific services only, see Listing 7.3.

```
<Policyset>
 <Policy>
  <Target>
   <Subject>localHomeDomain.[0-9a-zA-Z]*</Subject>
  </Target>
  <Rule permit>
   <Target>
    <Resources>
     <Resource>localHomeDomain.ownUserDomain.Service_1</Resource>
     [...]
     <Resource>localHomeDomain.ownUserDomain.Service_n</Resource>
    <Resources>
   </Target>
  </Rule>
 </Policy>
</Policyset>
```

Listing 7.3: Design pattern for a XACML Policy for public entities of the local Home Domain

Design Patterns for XACML Policies for Entities of other Homes

To express the different cases when entities that belong to other homes access a service that belongs to the own Domain (Cases 4-6 in Table 7.1), further design patterns are needed.

The SubjectID of a Policy used to control access of a Partner User Domain (Case 4) can be expressed as `[0-9a-zA-Z]*.UserDomain.[0-9a-zA-Z]*`. In this particular case neither the `HomeDomain` nor the `Entity`-part of the SubjectID contained in the XACML request do matter.

Policies controlling access of entities that belong to any User Domain of a Partner Home Domain (Case 5) have the Subject `HomeDomain.[0-9a-zA-Z]*.[0-9a-zA-Z]*`. In this case the `UserDomain` and `Entity`-part of the identifier do not matter.

Finally, Policies for public entities of a Partner Home Domain (Case 6) can be expressed as `HomeDomain.[0-9a-zA-Z]*`.

Design Pattern for Specific XACML Policies

Situations exist where the right to access a specific service needs to be granted to a specific entity. In such a situation the entire cryptographic identifier of the entity is used as SubjectID of the Policy and the entire cryptographic identifier of the service as ResourceID of the Policies Rule, see Listing 7.4.

```
1  <Policyset>
2   <Policy>
3    <Target>
4     <Subject>HomeDomain.UserDomain.Entity</Subject>
5    </Target>
6    <Rule permit>
7     <Target>
8      <Resource>HomeDomain.UserDomain.Service_1</Resource>
9     </Target>
10   </Rule>
11  </Policy>
12 </Policyset>
```

Listing 7.4: Design pattern for a XACML Policy for a specific entity

7.5 Guided Policy Administration

As already discussed, the manual creation or modification of XACML Policy Sets is a too complicated and error-prone process for inexperienced users. For this reason Domain Owners must be supported by an easy to use system especially adapted to the Domain scenario, which guides them through the process of policy generation.

Graphical Representation of Policy Sets

For human users graphical representations of abstract data are normally easy to comprehend. Luckily, the XACML Policies we discussed so far can be expressed graphically, as the basic nature of these policies is to "connect" the cryptographic identity of a Domain to the cryptographic identity of a service that this Domain may access.

As graphical representation a simple graph seems sufficient that connects a human readable identity of a Domain (e.g., the name of a Domain Owner) to one or several human readable service names. One edge in this graph corresponds to a single Policy, the entire graph to the Policy Set.

Vice versa, the graphical representation of access rights can be transferred into a valid XACML Policy Set. Based on this idea, a graphical policy editor can be created, which allows that even inexperienced Domain Owners specify or modify the XACML Policy Set of their Domain by simply "drawing" the appropriate edges in the graphical representation of access rights.

A slightly enhanced version of this very basic approach would be to decouple Domains and services by introducing roles. The Domain Owner can assign a role to a Domain in the similar fashion as just explained. Roles in turn give access to one or several services. An example of a Domain's access control graph is depicted in Figure 7.4. In this example several User or Home Domains are mapped to roles, which again are mapped to services.

The only role that exists per default in every Domain is the Domain Owner role, which is mapped to the "all" meta-service, i.e., to all services that exist in this Domain. The Domain Owner herself can specify further roles.

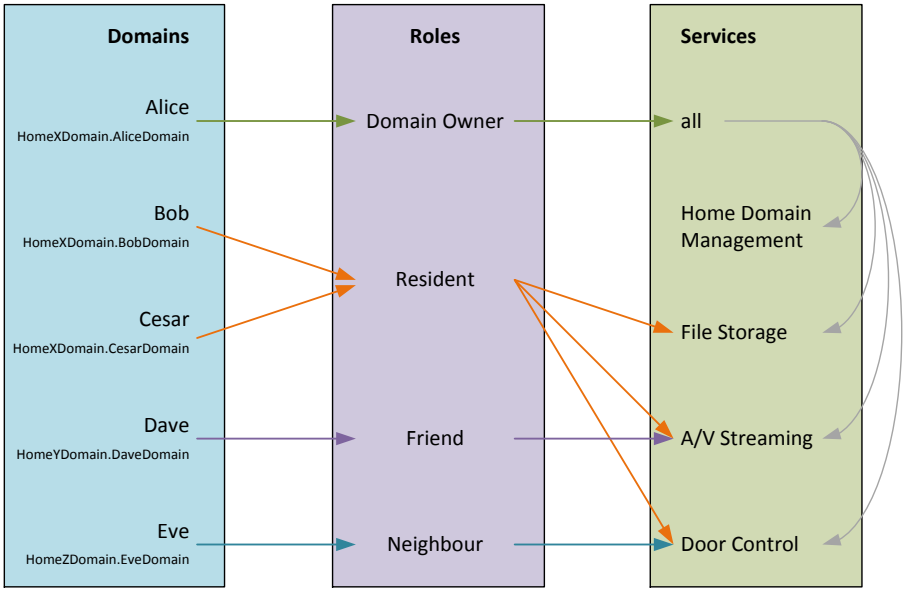


Figure 7.4: Example: Mapping between Users, Roles and Services

Database Representation

The Database of a Domain already has a `Domains table`, which contains information about friendly Domains, such as the human readable identifier (hID) of the Domain Owner, the cryptographic identity (cID), the certificate chain (certChain) and Identification and Reputation Levels (IL/RL). In order to reflect the roles a Domain *belongs to*, a new attribute "roles" must be added to the `Domains table`.

To reflect services and roles in the Domain DB, two new database tables are needed. The services existing in the Domain are represented in the Domain DB by the `Services table`. This table basically contains a human readable service name (hID) and the cryptographic identity (cID) of each service in the Domain. Roles are reflected in the database by the `Roles table`, which contains a human readable role name (hID) and references to services the members of a role *may access*.

An entity-relationship diagram that depicts the just explained database layout is shown in Figure 7.5.

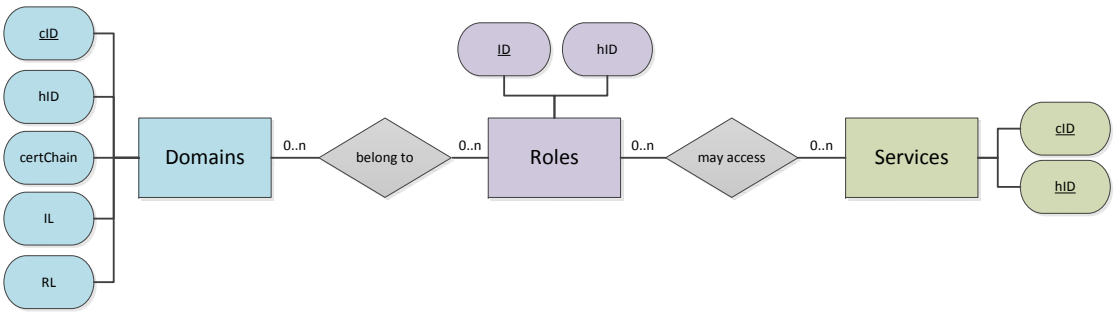


Figure 7.5: Entity-Relationship Diagram of the Domain Database

Translation of Domain DB into a XACML Policy Set

After the Domain Owner has specified the access control settings that should be used in her Domain, the Policy Generator is activated and transforms the content of the Domain DB to a valid XACML Policy Set. The transformation logic is described in Algorithm 7.1.

```
Data: Domains, Roles and Services tables from Domain DB
Result: XACML Policy Set
print <PolicySet>;
fetch Domains from Domains table;
foreach Domain in Domains do
    compute SubjectID for Policy;
    print <Policy><Target><Subject>SubjectID</Subject></Target>;
    fetch Roles of Domain from Domains table;
    foreach Role in Roles do
        print <Rule permit><Target><Resources>;
        fetch Services of Role from Roles table;
        if 'all' in Services then
            print <Resource>localHomeDomain.ownUserDomain.[0-9a-zA-Z]*
            </Resource>;
        end
        else
            foreach Service in Services do
                print <Resource>localHomeDomain.ownUserDomain.Service_n
                </Resource>;
            end
        end
        print </Resources></Target></Rule>;
    end
    print </Policy>;
end
print </PolicySet>;
```

Algorithm 7.1: Algorithm for Policy creation in a Domain

The algorithm first fetches all known Domains from the `Domains table` of the Domain DB. For each Domain, an own Policy will be added. According to the discussion in Section7.4 a suitable SubjectID is computed, which is then used as the Target of the Policy (`<Policy><Target><Subject>SubjectID</Subject></Target></Policy>`).

In the next step, the algorithm fetches the roles the currently processed Domain belongs to from the `Domains table`. For each role a new Rule (`<Rule permit><Target><Resources>...`) is added to the Policy. Now the Services that may be accessed by members of the currently processed Role are fetched from the `Roles table`. If the "all" meta-service is assigned to the Role, a "fuzzy" ResourceID is used to express that all services of the Domain may be accessed (`<Resource>localHomeDomain.ownUserDomain.[0-9a-zA-Z]*</Resource>`. Otherwise each service is added as an individual Resource (`<Resource> localHomeDomain.ownUserDomain.Service_n</Resource>`).

7.6 Feedback-based Learning of Access Rights

The PEP Proxy acts as intermediate between a service and the PDP of a Domain. Thus, all XACML requests and responses can be observed by this application. Besides logging of service access, this property enables another interesting feature of the proposed access control system.

When the PEP Proxy observes that a client tried to access a service but the request is denied by the PDP this might be caused by two reasons. First, because a client tried to access a service it is not intended to use. Second, because the Policy Set of the Domain is incomplete as the Domain Owner forgot to add the accessing Domain to a Role or did not know the cryptographic identity of a specific entity he wants to grant access to.

The PEP Proxy informs the Domain Owner, for instance, via the WebUI, about denied service requests, see Step 1 depicted in Figure 7.6. The Domain Owner can investigate this incident and decide if she desires to permit access to the Domain the next time (Step 2).

If the Domain Owner decides to grant the needed access right to a Domain, the PEP Proxy updates the Domain DB (Steps 3 - 4) and informs the Policy Generator about the modification (Step 5). Finally, the Policy Generator transforms the current Domain DB content into the corresponding XACML Policy Set (Steps 6 - 7).

The next time the client tries to access the service, the PDP will evaluate the XACML request using the updated Policy Set and permit access.

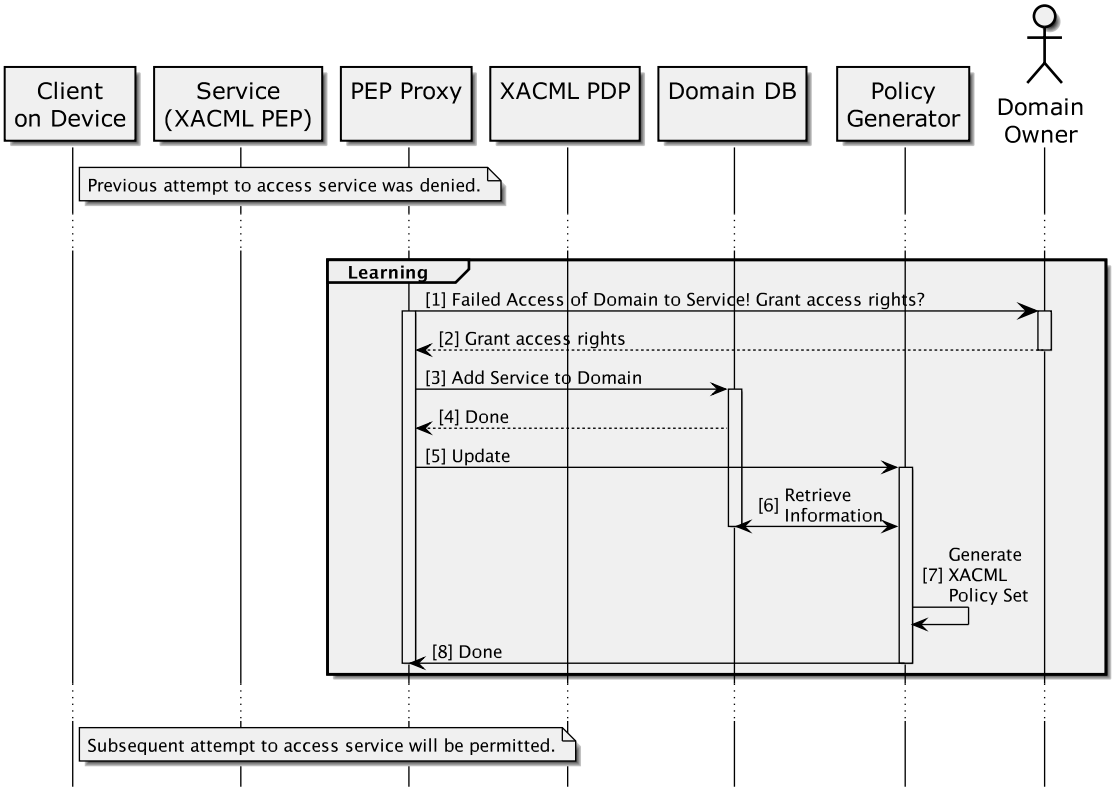


Figure 7.6: Feedback-based Learning of Access Rights

7.7 Discussion and Evaluation

7.7.1 Fulfillment of Requirements

The requirements defined on Access Control in Domains, see Section 7.1, could be fulfilled well by integrating the XACML authorization framework into the Guided Security Management System.

The TLS Handshake Interception, which is one of the two technologies required to integrate XACML to the existing system, is a quite elegant way how both legacy and newly

created services can be connected to the XACML PDP on one hand and use the cryptographic identities of accessing entities in XACML requests on the other hand. Furthermore, XACML policies use the cryptographic identities of devices and services to express access rights. By these reasons the integration of XACML authorization is fully compatible to the identification/authentication scheme proposed in this thesis (Requirement RB.4).

The design patterns of XACML policies for Domains presented in this chapter leverage the specific hierarchic structure of cryptographic identities to implement Domain-wise and Entity-wise access rights. Domain-wise access rights are implemented using "fuzzy" Policies that only test if an accessing entity belongs to an authorized Domain (RB.1). Entity-wise access rights are implemented based on Policies that match to the fully qualified cryptographic identifier of an entity (RB.2). Both, Domain-wise and Entity-wise Policies are able to grant access in fine granularity, i.e. to individual services and even to specific resources or actions offered by a service (RB.3).

The proposed Guided Policy Administration Point presents the access control settings of a Domain graphically to the Domain Owner. This representation is easy to comprehend by human users and also enables an inexperienced Domain Owner to implement her own access control concept with ease. As the Domain Owner does not have to edit the XACML Policies directly, in fact, XACML is entirely hidden from the Domain Owner, the Policy creation is also fault-resistant. This is because XACML Policy Sets generated by the Policy Generator are valid, meaning grammatically/syntactically correct. For this reason no disturbance caused by malformed XACML Policies are to expect (RB.6).

Lastly, the management of access control is centralized, i.e. policies are specified at a central place in each Domain (RB.5), which is quite convenient for the Domain Owner.

7.7.2 Experimental Proof of Concept

As proof of concept, we integrated XACML-based authorization using TLS Handshake Interception into various TLS-enabled Services.

The first service we integrated XACML-support into was a RADIUS server we used as a backend authentication service for the EAP-TLS-protected WLAN that gives access to the Service Network of a home. Normally, the RADIUS server would authenticate all devices that belong to a Domain if it is able to verify the presented device certificate. After integrating XACML into RADIUS it became possible to restrict WLAN access to *individual* devices of a Domain.

The *Devices Profile for Web Services (DPWS)* [46] offers functionalities comparable to UPnP but comes with built-in security features, i.e., offers client authentication using certificates. Based on DPWS, A/V streaming services can be implemented that offer better security than those services based on UPnP. Nevertheless, DPWS does not offer authorization. For this reason we integrated XACML into a DPWS-based A/V streaming service. From service requests we were able to extract names of the desired resource (server name and accessed file name) and the requested action. Using this information we were able to individually control access of specific entities to individual media files.

The third service we integrated XACML into was a simple, web-based control service for home appliances. Over a HTTPS-protected connection the user is able to access the service and trigger individual actions (e.g., lights on, lights off, display state). Again this information could be extracted and used for authorization.

7.7.3 Comparison to State of the Art and Related Work

The integration of XACML technology into the previously introduced identification/authentication scheme created a powerful access control system with user-friendly and secure

mechanisms for identity management and access control. The entire system does not only fulfill the requirements on access control in a future home environment we have defined, but is also highly competitive with the state of the art of access control mechanisms used in professionally managed environments.

An access control system quite comparable to the proposed system is Kerberos, see Section 3.4.1. As our system, Kerberos is able to authenticate entities that belong to Realms and authorize entities when they access a service. Our system is able to perform authorization in finer granularity as not only access to services but also access to individual resources and actions provided by a service can be authorized. Furthermore the management of our system is an integral part of the technology, can be performed by laymen, and does not depend on a highly skilled administrator.

Another often-used access control system in managed networks is RADIUS, see Section 3.4.2. As Kerberos, RADIUS provides authentication and authorization. Typically RADIUS is used in the context of network access control but add-ons for services exist, for instance, the RADIUS authentication module for Apache, which enable services to use RADIUS for authentication and authorization. As with Kerberos, the management of the system is no inherent part of the RADIUS system. Furthermore, the authorization performed by RADIUS also provides insufficient granularity, i.e., is only able to authorize per service.

Besides enterprise access control systems various research activities exist that target access control in home networks. Nokia Research has published a paper on usable access control inside home networks [11]. The proposed technology distributes identities, which are secret symmetric keys in this case, to devices using the Wi-Fi Protected Setup (WPS) technology and a registration system quite comparable to ours. Based on these identities, devices are authenticated and access to services is controlled. Nevertheless, this system has various drawbacks. 1) the administrative structure of the home is not reflected, i.e., a central authority in the home manages the *entire* network and *all* devices, 2) access control is only possible in coarse granularity, and 3) the system is based on the insecure PIN-based WPS, see Section 2.3.1.

Other works, such as [12, 13, 14], propose quite similar systems. The works leverage either a (modified) Kerberos server or another ticket-based Single Sign On system for authentication and authorization. As the previous work by Nokia Research, these works do not reflect on the home network structure and additionally do not answer sufficiently how users can operate the proposed system.

The authors of the Æther system [10] created a technology that has many conceptual similarities to ours. Æther authenticates an entity by a certified public key as our system does. The major difference between Æther and our system is that authorization is based in Æther on access rights encoded into a proprietary certificate format. This approach is quite comparable to SPKI/SDSI [47, 48]. The authorization granularity is fine and their system has the benefit over ours that no Policy Decision Point or comparable service has to be online to perform authorization. The drawback is that access rights encoded into a certificate are less flexible and simple to modify as access rights stored in a central Policy Set. Lastly, the proprietary certificate format prevents that Æther can be easily integrated into legacy software, which is quite unproblematic with our solution.

Another very important difference of our work to the mentioned related work is, that this work presents the only concept how access can be controlled *between* home networks. Other works solely focus on access control in the home. For this reason we are convinced, that this work presents the most comprehensive access control solution for home networking today.

The Guided Policy Administration Point proposed in this thesis is furthermore, to the knowledge of the Author, the only existing approach for XACML policy generation that abstracts over XACML syntax and additionally automates the implementation of access control settings in XACML syntax entirely.

7.7.4 Applicability

The named techniques were presented in the context of future home networks. Nevertheless, their applicability is not limited to this context, as all discussed concepts can be transferred to other environments. In a professional environment, our technologies can be used to integrate fine-grained authorization in existing services and to create a more decentralized management of access control, e.g., within subnetworks. If decentralized management is not desired, the proposed technologies help to disburden the central system administrator, as many management tasks, for instance the difficult and fault-prone generation of XACML Policies, are automated and greatly facilitated.

7.7.5 Open Issues and Future Work

The current implementation of the Guided Policy Administration Point is still quite simple and must be regarded as a proof of concept resp. a prototype. Future work will investigate how the editor can be extended in order to allow the Domain Owner to specify access rights in finer granularity as it is possible at this point in time, i.e., access rights to a specific resource or action a service provides.

For this purpose the `Roles table` of the Domain DB might be extended with additional attributes that reflect resources and actions. This extension will entail various further changes, for instance to the algorithm that translates the Domain DB to XACML policies. Nevertheless, the approach to generate XACML Policies from knowledge stored in a Database and the Domain Owner's settings expressed in a graphical way remains valid.

Another interesting extension of the policy editor might be functions that warn a Domain Owner if she grants the right to access a security critical service to a Domain that was not sufficiently identified.

For this purpose, the `Services table` might be extended with an attribute that expresses the criticality of each service that exists in the Domain. This information can be compared to the Identification Level of a Domain stored in the Domains table and, if the Identification Level is insufficient, the Domain Owner is warned.

Furthermore, the authorization system might be extended with location awareness. This extension can be based on a system we created that detects the presence of a user based on the visibility of a personal, portable Bluetooth-enabled device, e.g., a smart phone or a laptop, in a certain area. Depending on various external criteria (amount of deployed sensors, room size, material of walls, etc.), this system is able to detect in which room of a building the Bluetooth device/the user is.

This contextual information can be fed into the authorization system and, for instance, cause that a user may access a service only when she is present in the home or in the same room as the accessed resource. A possible use case for this technology might be that a user may only control the lights in the room she is currently in, for instance.

7.8 Conclusion

The mere authentication of entities is insufficient to control access in unmanaged networks, such as homes or small companies. For this reason, additional mechanisms are needed that

allow to implement fine-grained authorization policies, which control access to individual services and even specific resources or actions provided by a service, for instance.

XACML is a standardized authorization framework that offers the required flexibility and expressiveness for fine-grained authorization polices and is highly beneficial in unmanaged environments. However, in order to integrate XACML into the existing Guided Security Management System and into existing services two difficulties had to be solved.

The first difficulty is the technical integration of XACML into legacy services that exist in the home and to connect XACML-based authorization with the identification/authentication scheme we proposed earlier. This integration can be achieved by the TLS Handshake Interception technique we presented. Handshake interception basically hooks into the TLS handshake, extracts public keys of the presented certificate chain and computes the cryptographic identity of the accessing entity based on this information. This and the cryptographic identity of the accessed service are then both used to generate a XACML request, which is sent to and evaluated by the central XACML Policy Decision Point. Using TLS Handshake Interception every existing service that uses certificate-based authentication of the accessing client can be extended with XACML-based authorization.

The second difficulty is how the complicated and confusing XACML Policy Sets can be created and modified by inexperienced users. For this purpose a mechanism called Guided Policy Administration Point was proposed that basically displays the Domain's "knowledge" about other friendly Domains graphically. The Domain Owner can then assign services that may be accessed by entities of such Domains in a graphical way. Finally, a XACML Policy Set is automatically generated from the Domain's knowledge and the Domain Owner's settings. For the Domain Owner this is highly convenient as she is not bothered with the complex and confusing XACML syntax. Furthermore, the generated XACML Policy Set is valid, i.e., no disturbances caused by invalid (malformed) XACML Policy Sets need to be expected.

Finally, the specific hierarchical structure of the cryptographic identities we proposed earlier in the thesis is highly beneficial for access control. The hierarchical structure allows assigning access rights to individual entities, to entities of a specific User Domain and even to all entities that belong to an entire home.

Key Findings and Contributions

▷	Access control in home networks may not be solely based on authentication. Flexible and fine-grained authorization mechanisms are additionally needed.
▷	XACML is a suitable authorization framework that adds the needed feature to the home, but is too complicated to be used by inexperienced users.
C7.1	**TLS Handshake Interception** integrates XACML-based authorization into legacy services and connects XACML to our identification/authentication scheme.
C7.2	**Design patterns for Policy Sets** describe how an XACML Policy Set suitable for Home Domains resp. User Domains must be structured.
C7.3	The **Guided Policy Administration Point**, consisting of a **graphical policy editor** and a **Policy translation algorithm**, hides the complexity of XACML from the users and enables them to manage authorization rules of their Domains effortlessly and in a fault-proof manner.

Part III

Robustness of Security Components

8. Introduction to the Problem Area

The previously introduced Guided Security Management System for Domains enables average users to protect their unmanaged network based on enterprise grade security mechanisms. One problem that remains unsolved so far is how such users can be enabled to setup and maintain this system. The answer to this question is not trivial, as various difficulties exist.

First of all, the Guided Security Management System itself, especially the Registration Service, has several requirements on the topology and security of the local network. If these requirements are not fulfilled, the Registration Service cannot be used or its security is diminished.

Besides these infrastructural requirements, strong security requirements exist for the components of the Guided Security Management System. These requirements include that the authenticity and integrity of the components are guaranteed. Additionally, the confidentiality of the set of information (meta-data of Partner Domains, access control settings, etc.) that belongs to these components or the keying material they use must be assured. If the security of components and related information cannot be guaranteed, the security of the Domains that belong to the local network and of the whole local network is endangered. Security problems might finally lead to safety issues if the Guided Security Management System is deployed in a smart home or building where it manages access to functions that control doors and windows or the smart home's alarm system, for instance.

Another important problem is the resilience of the set of information that belongs to a Domain's Guided Security Management System. This is because this set of information describes all properties the Domain has, e.g., its identity (the Domain CA's key) or who is allowed to access which service (Domain DB). If this information is lost permanently the Domain is destroyed. In this case, the Domain Owner has no other choice than to recreate her Domain from scratch, which means to register entities, exchange trust, and create access rights again. This is obviously a cumbersome venture.

Even in managed network environments with professional system administrators some of the problems outlined above are quite difficult to resolve. But, as we have already discussed, typically no professional help is available in unmanaged networks. The users of such networks have too little experience in network and system administration and are not able to take care of the issues outlined above on their own.

8.1 The Domain Server

In the subsequent chapters, the architecture of a device called *Domain Server* and several *Auxiliary Services* are designed and evaluated. The Domain Server can be thought of as a next generation WLAN router. Besides providing the required network infrastructure, the device will act as a host for the Guided Security Management Systems of all Domains that belong to the local network. Auxiliary Services running on the Domain Server enable technically inexperienced users to setup and maintain the Guided Security Management System for their own Domain. Further Auxiliary Services will take care for the resilience of Domains and protect the security of the Guided Security Management System. To achieve the latter, various existing enterprise grade technologies, such as hardware virtualization (see Section 9.1), smart cards (see Section 9.2) and Trusted Computing technology (see Section 9.3) were examined and integrated into the Domain Server architecture.

Our approach is inspired by today's consumer electronics, such as WLAN routers or devices as Apple's Time Capsule. The beauty of these devices is that the user does not need to worry about the internal working or configuration of the device. The user basically only needs to buy the device, supply power, plug it into the local network, and is ready to use the offered services.

Despite being targeted to the smart home, the Domain Server or parts of the Domain Server functionalities are useful in other environments, e.g., in small company networks, as well. Also in managed networks, the Domain Server can be utilized to host the Guided Security Management System to further disburden the professional administrators.

8.2 High-Level Analysis and Requirements

The first step towards the Domain Server's design is to understand which high level requirements on its architecture exist and which Auxiliary Services are needed. Therefore, a first high-level analysis is performed hereinafter and overall requirements are defined. This first analysis serves primarily as an outlook. Each identified requirement will be elaborated in greater detail in following chapters.

Requirement A (RA): Secure and Flexible Service Hosting for Multiple Domains

From an economic point of view (hardware costs, power consumption) and also considering user expectations it is desirable that the Domain Server is able to host the Guided Security Management System of *all* Domains that belong to the local network. For instance, the Domain Server should host all Domains that belong to the same home or small company network. However, this multi-user mode may not reduce the security of Domains or reduce the flexibility of the overall system in any way. This requirement will be analyzed in greater detail in Chapter 10, where we also present and discuss a suitable solution.

Requirement B (RB): Provisioning and Separation of Networks

The Registration Service and other services have specific requirements regarding network security and topology. The Domain Server must provide a suitable network topology and thoroughly enforce the separation of networks in order to guarantee their security; see Chapter 10.

Requirement C (RC): Automation of Domain Management Tasks

The owners of Domains hosted on the Domain Server must be supported by Auxiliary Services that simplify or automate maintenance tasks, such as the creation of a new Domain. These Auxiliary Services need to be usable by inexperienced users, i.e., they must be transparent to the user, easy to comprehend, and easy to use; see Chapter 11.

Requirement D (RD): Availability and Resilience of Domains

Domains are defined by a set of information that reflects the Domain's identity or expresses who is allowed to access which service. The loss of this information set equals to the logical destruction of the Domain and causes major problems. Therefore, the *resilience* of this information set must be guaranteed to assure the Domain's resilience and the availability of all services controlled by the Guided Security Management System; see Chapter 11.

Requirement E (RE): Trust and Integrity of Execution Environments

The environment where the Guided Security Management System and related services are hosted/executed must have *integrity* and be *trustworthy*. This basically means that it must be excluded that unauthorized and potentially malicious software is present on the Domain Server. Such software might interfere with the Guided Security Management System or Auxiliary Services, diminish their security or even render them useless; see Chapter 12.

Requirement F (RF): Protection of Keying Material

Especially the private key used by a Domain CA to sign certificates of entities that belong to the Domain is a high value target for attackers and requires special protection. The prevention of key theft and other types of key abuse is a further overall requirement on the Domain Server; see Chapter 13.

9. Background

This chapter provides essential information about three technologies we used as main building blocks of the Domain Server.

Chapter Structure

Section 9.1 introduces hardware virtualization. Sections 9.2 and 9.3 are dedicated to hardware-based security mechanisms, namely smart cards and Trusted Computing technology.

9.1 Hardware Virtualization

Hardware virtualization (or in short: virtualization) is a technology able to create and execute one or several *virtual machines (VM)* on the same physical device. An individual operation system and any desired set of applications can be installed in each VM. For the user of a VM or an application installed on a VM there is almost no difference noticeable compared to using or being installed on a dedicated physical machine.

Hardware virtualization is today a frequently used technology. In companies virtualization is used to achieve higher infrastructural flexibility, to save costs for hardware, and as a security concept. In cloud computing environments virtualization is the basis for many offered services, such as Infrastructure as a Service.

Introduction and Discussion of Hardware Virtualization Technologies

A variety of open-source and commercial virtualization systems exist today. Xen [49] is currently one of the most popular and widely deployed open-source virtualization systems. Xen is used in industry and academia as it has the reputation of being efficient and secure regarding the effectiveness of isolation between VMs.

Another, widely deployed open-source virtualization system is the *kernel-based virtual machine*, which is publicly known as *KVM* [50]. KVM has also the reputation of being efficient, but security experts assess the isolation security between different VMs as lower compared to Xen [51, P. 11]. The main argument that supports this claim is the different architecture of both virtualization technologies. Xen is a so-called *type-1 hypervisor*, which means that VMs reside on top of a thin abstraction layer, the hypervisor, directly over the hardware. KVM, which is a type-2 hypervisor, uses a different approach. Here virtual machines and hypervisor reside on top of a complete Linux kernel over the hardware.

Security audits of the thin Xen hypervisor are easier to perform compared to audits of the voluminous combination of the KVM hypervisor plus the Linux kernel beneath it. For this reason Xen is expected to be less prone to errors and vulnerabilities than KVM and to offer better isolation security between VMs.

Xen Terminology

In Xen terminology, the physical machine itself is denoted as *host*, VMs are called *guests*. The containers that isolate guests are referred to as *Xen domains*[1].

Xen differentiates between one privileged *domain zero (dom0)* and multiple unprivileged *user domains (domU)*. The privileged dom0 has a special position in the Xen virtualization system, as it is possible to control the physical host, the hypervisor, and domUs from this place. For this reason, a user that has access to dom0 or an application installed within dom0 can create, start, stop, and destroy guests or may modify settings considering the internal networking of the system, for instance.

Data that belongs to a guest, e.g., operation system, applications, and user data, is typically stored inside a *disk image*. Disk images are large binary files typically stored on the file system of the host. Alternatively, *Logical Volume Manager (LVM)* [52] can be used to assign a so-called *logical volume* to a guest where the guest's data can be stored in. Regardless which technology is used we use the term *disk image* or simply *image* to refer to the data that belongs to a guest.

Virtual Appliances

A *virtual appliance* is a preconfigured VM image that contains an operation system and ready to use applications. The user does not need to create a VM, install an operation system and applications, and finally configure the installed software. Instead, the user simply downloads a virtual appliance to the physical host where the virtual appliance can be started as VM and be used immediately.

Today, the usage of virtual appliances is quite common in cloud computing environments, such as Amazon EC2. Here numerous virtual appliances[2] exist that were created either by the cloud operator or voluntary maintainers.

9.2 Smart Cards

Smart cards are person-bound, highly portable devices that are used for a wide range of purposes. Depending on the purpose, different smart card technologies are used. The most widely deployed card technology are cryptographic token, which are used for the authentication of an user or digital signing purposes. Other use cases may require more flexibility and more sophisticated smart cards. Such smart cards contain tiny applications, which are executed on the card. For this reason this card type is called *processor card*. One example where this technology is used is it the *Geldkarte*, a technology popular in Germany that implements a digital wallet.

The overall benefit of smart cards is the high level of security provided by this technology. The chips on smart cards are designed to be tamper-proof. This means they withstand unauthorized modification or espionage on data (keys, certificates, applications) stored on the card even by an attacker that has physical access. Furthermore, smart cards, no matter which technology is used, provide a secured execution environment for critical computations that is isolated from the computer system where the smart card is plugged into. By these reasons, smart cards are typically highly secure and trustworthy and also applications that leverage this technology are typically secure.

[1] Please note: Xen domains may not be confused with the Domain concept introduced in this thesis.
[2] Amazon denotes virtual appliances as Amazon Machine Images (AMI).

9.2.1 Cryptographic Token

Cryptographic token are able to generate and store asymmetric key pairs. The keys generated within a cryptographic token cannot be extracted and do never leave the token as the card itself performs computations that involve the key. Alternative to generating keys inside the smart card, it is possible to import existing private keys generated by software, e.g., by OpenSSL.

The communication with cryptographic token and other simple smart cards, e.g., older Java smart cards, is usually based on *Application Protocol Data Units (APDU)*, which are specified in the ISO standard 7816-4 [53]. APDUs describe the format of messages used by a simple request/response protocol on layer 7 of the ISO/OSI model. For APDUs, the transport over lower ISO/OSI layers is transparent, classic cards with contacts, contactless cards, or even cards that are integrated into an USB dongle use the same APDU format.

Besides the format of APDUs, the communication with cards on lower layers of the ISO/OSI model is underspecified. Each card manufacturer or even each card model uses a different set of instruction or response codes. Several libraries, such as *OpenSC* [54], provide an abstraction over the low-level communication specific to different smart cards models.

Furthermore, RSA has specified vendor and card independent standards for programming interfaces (API) to cryptographic token known as PKCS #11 [55] and PKCS #15 [56]. Both facilitate the usage of cryptographic token by applications. PKCS #11 gives easy access to functions like signing and decrypting data, listing keys and certificates, PIN handling and so on. PKCS #15 provides further functions related to card management and formats of data structures.

9.2.2 Java Smart Cards

Processor cards are tiny computing systems equipped with CPU, main memory, persistent storage, and a simplistic operation system. Small applications, which are typically called *applets*, can be installed on and executed inside a processor card isolated from the computer system the card is currently plugged into.

Java smart cards are one specific example for this technology. Applets for such cards are written in *JavaCard*, a minimalistic Java platform. Compared to standard Java platforms for computers or larger embedded systems, JavaCard only supports a limited subset of the Java language and not all defined types.

Currently two versions of the JavaCard standard exist and are in use: the older version 2 [57], which dates back to 1996, and the newer version 3 [58], which was standardized in 2008. Version 3 of the JavaCard standard describes two different flavors of the same technology with distinct features.

JavaCard Version 3 *"Classic Edition"* [59] is only a modest evolution of Version 2 of the standard. JavaCard Version 3 *"Connected Edition"* [60] must be considered as a revolutionary step in smart card technology. The most fundamental change is the new option to communication with Java applets using HTTP over TCP/IP. Optionally, the security of HTTP communication can be guaranteed by TLS/SSL. A further benefit is the widely enhanced Java support that offers a greater variety of functions, types and libraries. Finally, the new standard dictates enhanced hardware properties. Smart card hardware for Connected Edition JavaCard comprises a 32-bit CPU, 24kB RAM and at least 128kB EEPROM. Older card hardware typically includes only a 8- or 16-bit CPU, about 2KB of RAM, and a 8 to 32KB EEPROM [58, P. 4].

At the time of writing this thesis no Java smart cards that support Version 3 of the standard in Connected Edition could be purchased. For development purposes, a card simulator that runs a reference implementation of the new standard can be used. This simulator is included in the Windows version of NetBeans IDE.

9.3 Trusted Computing

The term *Trusted Computing* was first used in the *Trusted Computer System Evaluation Criteria (TCSEC)* [61] standardized by the US Department of Defense in 1983. The purpose of TCSEC was to provide a standard for the classification and evaluation of the effectiveness of different security mechanisms built into computing systems. In 2005 TCSEC was used as the basis for the *Common Criteria for Information Technology Security Evaluation* [62], which are today the international standard for assessment and certification of computer systems.

In 1999, the *Trusted Computing Platform Alliance (TCPA)*, a consortium consisting of companies such as Microsoft, IBM and Hewlett Packard was founded. TCPA's aim was to design an industry standard for trustworthy computing systems for civilian use. TCPA and Trusted Computing gained large public attendance in 2002 after Microsoft announced features of its Trusted Computing operation system codenamed *Palladium*. Palladium was described as *"[...] an evolutionary set of features for the Windows operating system. Combined with a new breed of hardware and applications, these features will give [...] greater data security, personal privacy, and system integrity."* [63]

IT experts understood these features as basis for an operating system that withdraws control over the computer from the user. Security experts, such as Ron Rivest, Bruce Schneier, or Richard Stallman published negative articles [64] and a counter movement against TCPA and Palladium was founded. Despite the efforts of TCPA to rebut [65] these articles, Palladium and Trusted Computing were publicly refused. For this reason TCPA was closed in 2003.

In the same year, former TCPA members founded the *Trusted Computing Group (TCG)*[66], which still exists today. The working groups of TCG still have the objective to design an industry standard for Trusted Computing systems.

The quite negative image Trusted Computing technology received in the past still sticks to the TCG and Trusted Computing to some extent. This might be the reason why this technology is not used more often today. Seen in a sober light, Trusted Computing is, according to the Author's opinion, a powerful set of technologies that can be used for the good or the bad. Besides questionable technologies, such as DRM (Digital Rights Management), also undoubtedly important technologies that protect a computer system and its user can be based on Trusted Computing.

9.3.1 Definition of Integrity and Trust

Before the TPM and Trusted Computing technologies are introduced, it is important to define the terms *integrity* and *trust* in the context of a computer system.

Integrity

The term integrity has different meanings in different contexts. In computer or network security *data integrity* of a file or a network packet, for instance, is ensured if an unauthorized party cannot modify this data without being detected. *Program integrity* differs from this definition slightly. Sailer et al. [67] define the term as follows: *"The integrity of a program is a binary property that indicates whether the program and/or its environment*

have been modified in an unauthorized manner. Such an unauthorized modification may result in incorrect or malicious behavior by the program, such that it would be unwise [...] to rely on it."

This definition requires some further explanation. First, the integrity of a program cannot be viewed in isolation from the *"environment of the program"*, which is the operation system, shared libraries, configuration files, etc. So a program cannot have integrity, if the computer system that executes this program does not have integrity as well. This is a difference to the definition of data integrity where the environment the data is processed in does not play any role. Therefore, program integrity can be basically equated to computer system integrity. Sailer also describes that modifications *"in an unauthorized manner"* to the program's binary file (*"binary property"*) will violate the program's integrity. With this understanding, program and system integrity become measurable and verifiable using the same cryptographic operations (hashing, signing, etc.) that can be used to measure and verify data integrity.

Finally, Sailer et al. regard program integrity as a requirement that must be fulfilled so that a user is able *"to rely on"* the program, the computer system and all programs executed on it.

Trust

A definition of the term *trust* is given by TCG in the context of a computing device as follows: *"Trust is the expectation that a device will behave in a particular manner for a specific purpose."* [68] This means that the device, e.g., a computer system, will perform the tasks it is supposed to do in the intended way.

If the definitions of trust and integrity are combined, then trust in a computer system can be understood as a measurable, binary property that requires operation system, shared libraries and all programs to have integrity.

9.3.2 The Trusted Platform Module

The *Trusted Platform Module (TPM)* is a cryptographic chip standardized by the TCG [69, 70, 71]. As the TPM is low-cost it can be integrated into devices without increasing the overall costs much. Therefore, many hardware manufacturers, especially those who are members of the TCG, ship their products already with TPMs.

The TPM is the key hardware component of the Trusted Computing paradigm. The core idea of TCG was to provide a piece of trustworthy hardware, a *trust anchor*, for computer systems able to execute cryptographic functions isolated from the remaining computer system. According to the TCG, the TPM is resistant to software-based attacks, as the chip follows strict rules how data is internally processed and which functions can be called from the outside. In TCG terminology, a computing system that complies with TCG specifications and that is equipped with a TPM, is denoted as a *Trusted Computing Platform (TCP)*.

In this thesis two TPM functions are especially relevant. The first functionality is key handling, i.e., the generation, storage and usage of asymmetric keys. The second important functionality is the TPM's ability to assist applications executed on the TCP that measure and attest the integrity of this platform.

The latest public version of the TPM standard is 1.2, which is also introduced in the following. However, the standardization process of the TPM version 2.0 is already underway. No details of the new standard are published yet, but it can be expected that the new standard will not bring fundamental changes but enhancements to existing TPM functionalities. For instance, new or improved cryptographic hash and encryption algorithms might be included into the upcoming standard to make the TPM 2.0 future-proof.

9.3.2.1 Functional Building Blocks

The TPM chip comprises of three major functional building blocks:

Non-volatile memory:

Non-volatile memory stores the *Endorsement Key (EK)* and the *Storage Root Key (SRK)* persistently within the TPM. The EK is the unique TPM and TCP identity. Its purposes are detailed in Section 9.3.2.3. The SRK is the root key of the TPM key storage functionality, which encrypts keys with the SRK and stores them safely outside the TPM. Both keys are most relevant to the TPM and platform security. Therefore, non-volatile memory offers protection to keys, which means that an adversary cannot extract keys from the TPM without manipulating the hardware.

Volatile memory:

Keys apart from EK and SRK are encrypted using the SRK and stored outside the TPM on the TCP's persistent memory (e.g., hard drive). When such a key should be used, the still encrypted key is fetched into the TPM, decrypted, and used in the volatile memory of the TPM. A different example for volatile memory are *Platform Configuration Registers (PCR)*, which are detailed in Section 9.3.2.4. Similar protection as for non-volatile memory is offered.

Functional unit:

The functional unit is able to compute critical cryptographic operations within the shielded TPM environment in isolation from the remaining computer system. Examples for such operations include the generation of asymmetric or symmetric keys or the calculation of signatures. Again it is impossible for an attacker to interfere with these operations without manipulating the hardware.

9.3.2.2 Key Types

The TCG has defined several key types with different properties (restrictions). The properties of a key are defined when the key is generated and later enforced in hardware by the TPM. Once the generation of a key is finished, its properties cannot be altered anymore.

One of the most fundamental properties of a private key residing in a TPM is the ability to be *migratable* or not (*non-migratable*). Migratable private keys can be exported from one TPM and be imported into another TPM. Non-migratable keys are guaranteed to never leave the TPM. Public parts of all TPM key types can be exported from the TPM and be given to external entities, which may use this public key to verify signatures created by the corresponding private key, etc.

The TCG differentiates between two overall key classes: *signing keys*, which are asymmetric keys used to sign data, e.g., files or messages. The second class are *storage keys*, which are asymmetric keys used to encrypt data, e.g., symmetric keys.

- **Endorsement Key (EK):** The EK is a signing key typically created during the manufacturing process of a TCP. The EK must be considered as the unique identity of the TPM and the TCP. Therefore it is one of the most relevant TPM keys. Due to privacy concerns, the EK is never used for encryption or signing of user data but only in the creation process of AIK Credentials (see section 9.3.2.3).

- **Attested Identity Keys (AIK)** are user-generated, non-migratable signing keys that are used to sign data contained within the TPM. For instance, the AIK signs a *TPM_QUOTE_INFO* data structure [70, P. 96], which contains the content of one or several *Platform Configuration Registers (PCR)* (see Section 9.3.2.4). A different example of AIK usage is the signing of data structures that contain public parts of other, non-migratable TPM keys. These so called *TPM_CERTIFY_INFO* data structures [70, P. 99] additionally give verifiable evidence of type and properties of a certified key.

- **Bind and Sealing Keys** are non-migratable storage keys and are used to encrypt small portions of data, for instance, symmetric keys. These symmetric keys may be used to encrypt larger portions of data, a file, for instance. Both key types guarantee that encrypted data can only be decrypted on the same TCP. Sealing keys additionally guarantee that this data can only be decrypted if the TCP is in a specific configuration, i.e., if specific values are contained within the TPM's PCRs.

- **Signing Keys** are used to sign arbitrary data, which is also allowed to reside outside the TPM, e.g., a file stored on the TCP's hard drive or protocol messages. Signing keys can be migratable or non-migratable.

- **Legacy Keys** are asymmetric keys generated in software, e.g., by OpenSSL, and later imported into a TPM. Legacy keys are migratable by default.

9.3.2.3 Trusted Computing Platform Credentials

A TCP is useless if the user of this platform or a different computer system that communicates with this platform cannot verify whether this platform is equipped with standard compliant Trusted Computing technology or not. For establishing trust into a TCP, TCG has specified certified data structures, so called *credentials*, which are directly or indirectly bound to the TCP or, more precisely, to the Endorsement Key of the platform's TPM.

Endorsement Credential:

The Endorsement Credential contains the public part of the EK and expresses that this EK resides in a standard compliant TPM chip. EK and EK credential are typically created by the manufacturer of a TCP as part of the manufacturing process.

Conformance Credential:

The Conformance Credential expresses that a platform type, e.g., a certain notebook series, complies with common evaluation guidelines including that the platform is equipped with a standard compliant *Core Root of Trust for Measurement (CRTM)*, see 9.3.2.4. The Conformance Credential does not contain any reference to an individual specific platform.

Platform Credential:

The Platform Credential is typically created by the manufacturer and contains references to the EK Credential of the TCP's TPM and the Conformance Credential of the platform. This credential's purpose is to connect a specific platform to a Conformance Credential, i.e., it expresses that a certain platform identified by its EK is a TCP.

Attestation Identity Credential:

The AIK credential contains the public part of an AIK and is issued by an authority called *Privacy CA*. The Privacy CA issues an AIK credential after verifying the signature created by the TPM using its EK of the public part of this AIK. Additionally, the just explained platform credentials are evaluated to establish trust into the TPM and the TCP.

Thus, an AIK credential expresses that a specific AIK resides within a standard compliant TPM chip integrated into a standard compliant TCP. The benefit of this Credential is that an entity that receives a signature created by an AIK and the corresponding AIK Credential is able to establish trust into the TPM/TCP but cannot link the AIK to a specific TPM/TCP.

9.3.2.4 Integrity Measurement, Reporting and Verification

One of the most important concepts of Trusted Computing is the trustworthy measurement and reporting of system integrity. The term *measuring* describes the creation of SHA-1 *fingerprints* of software components executed on the TCP and the subsequent storage of fingerprints within the TPM's Platform Configuration Registers (PCR). Typically, the measurement process takes place during the boot process of the TCP. For this reason, TCG refers to this process as *Trusted Boot*.

One or several fingerprints stored in the TPM's PCRs can be signed later using an AIK. TCG calls this *quote operation* and the resulting signed data structure *TPM_QUOTE_INFO*. For simplicity we refer to this data structure later as *Validation Structure*. Signature, fingerprints and AIK credential are sent to an external entity, which then may verify the received information. These processes are known as *Remote Attestation* and *Integrity Verification*.

After this short overview, the introduced concepts are detailed.

Platform Configuration Registers (PCR)

A Platform Configuration Register is a 160 bit wide protected memory area inside the volatile memory of the TPM. The TPM guarantees that a PCR cannot be reset (set to zero) or set in a targeted manner to a specific value on request by a user after the PCR has been set initially.

The only option explicitly allowed to modify an initialized PCR is known as *extending* the PCR. The `extend` operation first concatenates ($|$) another fingerprint (n) to the current content of PCR i (PCR_i). Then the SHA-1 hash function is applied to the concatenated value. The result of the `extend` operation is again stored in PCR i.

$$PCR_i = SHA1(PCR_i|n)$$

Further exceptions from this rule are "dynamic" PCRs. An application is allowed to set and reset dynamic PCRs on purpose. As each PCR, both "ordinary" and dynamic, have individual numbers, PCR types can be easily distinguished.

The Core Root of Trust for Measurement

The TPM itself is passive and not capable to measure the integrity of any system component. For this reason, TCG has specified the *Core Root of Trust for Measurement (CRTM)* that creates the measurement values of system components executed at early phases of the boot process, i.e., the system's BIOS and boot loader.

The CRTM is a piece of executable code typically stored within the BIOS firmware. The trust into the correct functioning of the CRTM is established using the Conformance Credential described in Section 9.3.2.3.

Creating a Transitive Chain of Measurements

The CRTM is only able to measure system components loaded early in the boot process. A special TPM-enabled boot loader, called *Trusted Boot Loader*, extends the transitive chain of integrity measurements up to the system kernel and kernel modules. One example for a Trusted Boot Loader is *TrustedGRUB* [72].

Sailer et al. present in [67] a mechanism called *Integrity Measurement Architecture (IMA)*, which is a patch for the Linux kernel that extends the transitive chain of integrity measurements up to the user space and the system's run time. The combination of CRTM, Trusted Boot Loader and IMA is therefore able to measure and log fingerprints of all system components loaded and executed on a computer system. The "extended" Trusted Boot process is depicted in Figure 9.1.

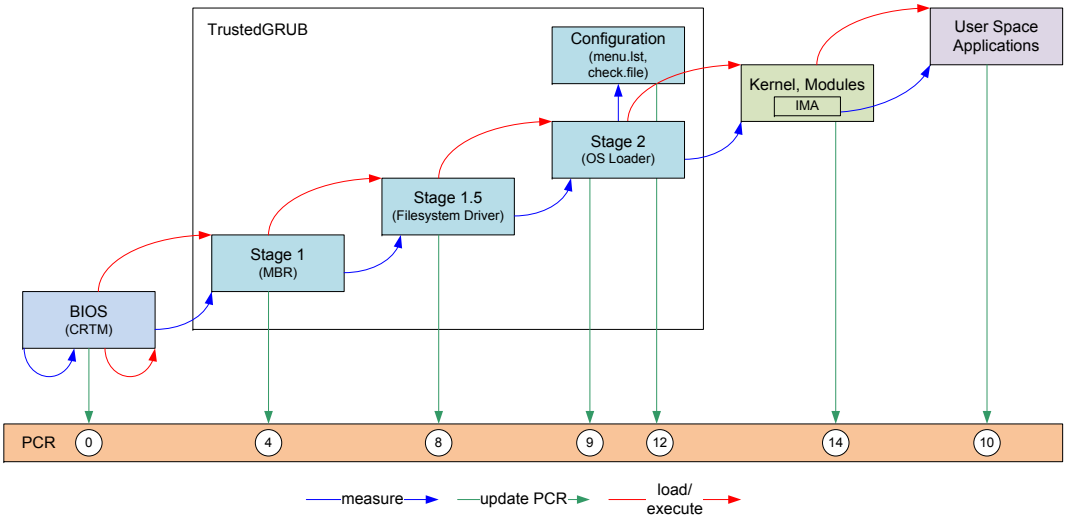


Figure 9.1: "Extended" Trusted Boot process, taken from [73].

As the TPM does not offer enough PCRs to store all fingerprints created by IMA, IMA stores fingerprints in an *Integrity Measurement List (IML)*. The IML is basically a plain text file and requires protection against unauthorized modifications by an attacker. This protection is realized by computing a recursive checksum, called *Integrity Verification Value (IVV)*, using the `extend` operation described above. When an application is loaded, IMA first creates a new fingerprint, extends this fingerprint to PCR 10 and logs the fingerprint to the IML. After this process is finished, IMA loads and executes the application. Listing 9.1 shows an extract of an IML. The first column expresses into which PCR the SHA-1 fingerprint (second column) was extended. The third column is the path to the loaded/executed application or library.

```
1  10 9797edf8d0eed36b1cf92547816051c8af4e45ee boot_aggregate
2  10 5cb60e502db7b364f5057fa998d827abf3b2d14b /bin/sh
3  10 e5c8719a2506e43b80ebf850ccdfd4d7a42e3a10 /sbin/depmod
4  ...
5  10 d61732d3496779773062218e7ccb9f8601ce1a64 /usr/lib/firefox/firefox-bin
6  10 bff09fe7dfb4dd7d2be20f698be0589e4519f508 /usr/lib/firefox/libmozjs.so
7  10 11f56dfd57be7b24cc237b337e78324b6c2bd3d9 /usr/lib/firefox/libxpcom.so
8  ...
```

Listing 9.1: Example: Integrity Measurement List (IML)

The already described behavior of PCRs makes it impossible for an attacker to modify the IML without being detected, for instance, to delete the fingerprint of an unauthorized application. This is because the attacker is unable to set PCR 10 to an IVV that matches to the IML.

Remote Attestation and Integrity Verification

In order to prove the integrity of a TCP to another entity, typically called *verifier*, a protocol known as *Remote Attestation* is used. For Remote Attestation, the verifier first provides a random nonce to the TCP, which is signed together with the content of one or several PCRs using the TPM `quote` function. The TPM uses for the creation of this signature an AIK.

The result of this operation, the signed Validation Structure, is sent together with the AIK Credential to the verifier. The verifier now checks AIK credential and signature for validity and freshness. Now the verifier compares the reported fingerprints to expected trustworthy fingerprints deposited in a list of authorized and trustworthy software components ("white list"). If all checks are successful, the integrity of a computer system could be successfully verified.

If IMA is used on the computer system that should be verified, the verifier will receive additionally an IVV and the IML. Two additional steps are now required for integrity verification. The verifier must first compute the checksum over the IML itself and compare the result to the received IVV. If both IVVs match, the IML can be trusted. Second, the verifier needs to compare each fingerprint contained in IML to its white list. If these checks are completed, the verifier may trust the remotely verified computer system.

10. Domain Server Architecture

As we explained before, the envisaged Domain Server can be imagined as next generation WLAN router that provides the required network infrastructure and hosts the Guided Security Management Systems of all Domains that belong to the local network securely. Additional Auxiliary Services executed on the Domain Server are needed to help the inexperienced users to setup and maintain their personal Domain's Guided Security Management System.

The Domain Server's basic architecture is mostly influenced by the first two overall requirements we have defined in Chapter 8, i.e., "Secure and Flexible Service Hosting for Multiple Domains" (RA) and "Provisioning and Separation of Functional Networks" (RB). For this reason we address both requirements in this chapter. As the basic design of the Domain Server will also influence the solutions that address the remaining overall requirements, these requirements are partially discussed in this chapter briefly as well.

Chapter Structure

After a refined problem analysis in the subsequent Section 10.1, fine-grained requirements on the solution are defined in Section 10.2. The design of the Domain Server architecture is explained in Section 10.3 and discussed in Section 10.4. Finally the chapter is concluded in Section 10.5.

Acknowledgments

This chapter contains results developed by Simon Stauber and Simon Mittelberger in their work as student research assistants. These works were assigned and supervised by the Author in the context of his doctorate.

10.1 Refined Analysis

In this section, a refined analysis of both overall requirements (RA and RB) addressed in this chapter is performed.

Secure and Flexible Service Hosting for Multiple Domains

Hosting the Guided Security Management Systems and other security relevant services (e.g., XACML Policy Decision Point) of *several* Domains on the *same* Domain Server does

not appear to be a big technical problem at first sight. For the sake of brevity we refer in the following to all services related to the security of a Domain as the Domain's *Security Services*. But, as we pointed out before, these Security Services are highly security sensitive and if no suitable precautions are taken hosting Security Services of several Domains on the same Domain Server might result in various problems that finally reduce the security and effectiveness of those services. In the following, possible negative impacts caused by the required multi-domain/multi-user mode are analyzed in order to understand which measures need to be taken to provide a high level of security and also flexibility.

The different Domains hosted on the Domain Server or their owners must be considered as different stakeholders with different and possibly conflicting interests. So it is possible that one stakeholder, either willingly or accidentally, harms another stakeholder's Domain. This harm might be caused, for instance, by tampering with the set of information that belongs to a Domain (keys, received certificates of Partner Domains, etc.), by extracting keying material from a Domain CA, or by exchanging an authentic security component with a modified, bogus version. It is a widely used best practice in system design to separate different stakeholders from each other to prevent such problems. Therefore, the separation of Security Services of different stakeholders on the Domain Server is a first and important requirement to secure the envisaged multi-user mode.

Besides her Domain's Security Services, a Domain Owner might want to host *Additional Services* on the Domain Server. As Additional Service a service is understood that is not related to Domain security, e.g., network attached storage (NAS), A/V streaming, home control, etc. These Additional Services originate from third party software manufacturers and might contain malware, such as back doors, rootkits and spyware. Consequently, Additional Services pose a threat at the Security Services hosted on the Domain Server. Therefore a strict separation between Security Services and Additional Services is required as well.

On the Domain Server another, non-obviously visible stakeholder is present. This stakeholder is no human user but the Auxiliary Services required to provide help for inexperienced users of the Domain Server to setup and maintain Domains (see Chapter 11) and functions that control/enforce the system's integrity (see Chapter 12). As these services have high influence on the security of all Security Services and the whole network they must be protected and run isolated from all other services on the Domain Server.

Provisioning and Separation of Functional Networks

In Figure 10.1, a section of an example network X, which might be a home or small company network, and a User Domain X.A that belongs to this network is depicted. The gray boxes symbolize the Security Services (box on top) and Additional Services (box on bottom) that belong to Domain X.A that need to be hosted on the Domain Server (large, light gray box). Some components of the Security Services and some Additional Services need to be connected to the outside of the Domain Server in order to perform their duties. These connections must be reflected by the internal network architecture of the Domain Server.

The Registration Service requires the first interface to the outside of the Domain Server. As already outlined in Chapter 5 we proposed to utilize an open (unprotected) WLAN, called *Registration WLAN*, to connect an unregistered device to the Registration Service of a Domain. As all devices are able to access the open Registration WLAN, no services except those related to registration may be accessible from the Registration WLAN.

After registration, a device possesses a valid certificate for her private key signed by a Domain's CA. We already explained that wireless devices may utilize this certificate to

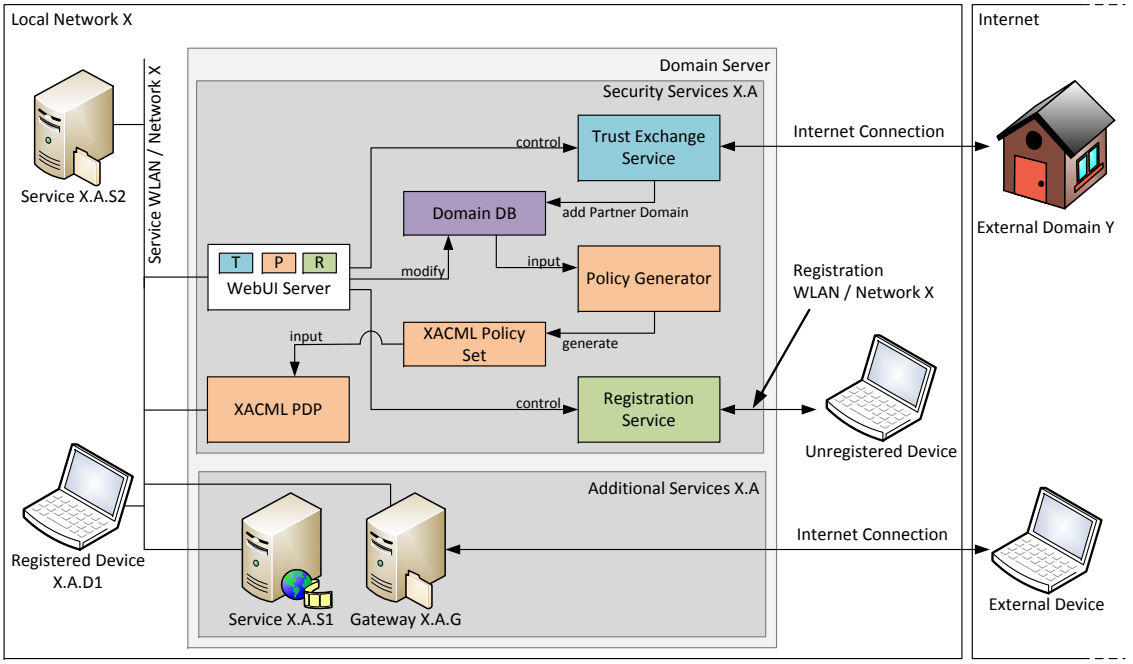


Figure 10.1: Example: Section of an example local network topology with various involved entities.

authenticate to a second, EAP-TLS protected WLAN, called Service WLAN. The Service WLAN is used to provide connectivity to the internal home/company network, called Service Network. As the name suggests, the Service Network interconnects all devices and services of all Domains that belong to the local network. For instance, the registered device X.A.D1 connected to the Service Network may connect to the WebUI component of Domain X.A's Guided Security Management System in order to control a device registration process or to modify the Domain's access control settings. This device and other devices must also be able to connect to local services, such as X.A.S1 hosted on the Domain Server or X.A.S2 hosted on some other device in the network. Finally all services may communicate via the Service Network and need to connect to the Domain's PDP in order to request authorization decisions in case a device wants to access the service. The Internet Trust Exchange service is a special case as this service must be directly reachable from the Internet in order to work.

So far, requirements on the internal networking were discussed only. Additional requirements arise from external networking, i.e., from the need that devices located at the outside of the local network may remotely connect to the inside of the local network in order to use local services. Devices located in the public Internet or in other networks are examples. For this purpose, a gateway service, e.g., a VPN server (X.A.G), is needed, which must be publicly reachable from the Internet.

10.2 Requirements

After analyzing properties of the Security Services, of the network topology, and of other needed functionalities the following requirements are defined on the basic Domain Server architecture.

Secure and Flexible Service Hosting for Multiple Domains

- **RA.1: Separation of Security Services of Different Domains:** The Security Services of different Domains need to be strictly separated.

- **RA.2: Separation of Security Services and Additional Services:** Security Services of a Domain must be strictly separated from Additional Services.
- **RA.3: Separation of Auxiliary Services and all other services:** The highly critical Auxiliary Services that control the Domain Server itself must be strictly separated from the rest of the services located on the Domain Server.
- **RA.4: Secure Separation/Isolation:** The separation mechanism must be secure, e.g., withstand attacks by malware or human adversaries, in order to guarantee the effectiveness of the protection.
- **RA.5: Flexibility:** The solution that separates Domains must be flexible, i.e., provide the same amount of usefulness as separate dedicated servers would.

Provisioning and Separation of Functional Networks

- **RB.1: Provisioning and Separation of Service and Registration Network:** The Domain Server must enforce a strict separation of Registration Network (unregistered entities) and Service Network (registered entities).
- **RB.2: Direct Internet Connectivity for Trust Exchange and Gateway Services:** The Internet Trust Exchange and gateway services need to be directly reachable from the Internet.

10.3 Design

This section discusses design options for the Domain Server's basic architecture and presents the final design.

10.3.1 Options for Stakeholder Separation

The required separation of Security Services that belong to different Domains, between different service classes (Security Services/Additional Services), and of the Domain Server's auxiliary functionalities from the rest of the system can be achieved based on various existing technologies.

Per se, Linux is already a multi-user operation system and capable to isolate processes and control access to data that belongs to different users. Access is typically controlled based on read-write-execute permissions assigned to users. This means that each user (or an application/service owned by a user) possesses a specific set of access rights that control which data the user is allowed to access or which system maintenance tasks the user is allowed to execute. So a basic solution might be to create separate user accounts, one for each stakeholder's Security Services, one for each stakeholder's Additional Services and one for the Auxiliary Services that control the Domain Server. User accounts assigned to stakeholders will obtain standard (limited) user permissions; the Auxiliary Service will obtain full (root) access in order to be able to perform system maintenance. Unfortunately this solution is not sufficient due to various reasons.

First of all, software vulnerabilities can lead to so called privilege escalation attacks that allow an unprivileged user to obtain root access. Once this user got hold on root privileges, she is able to access and modify everything on the system, also data and configuration files that belong to other Domains. So the isolation strength between users provided by standard Linux must be regarded as insufficient.

There exist countless software and hardware-based approaches that address this issue by attempting to harden the Linux OS. However, hardening Linux is still not enough, as

the Linux access control model does not offer enough flexibility. This lack of flexibility might reduce the usefulness of the Domain Server and even leads to new security problems. For instance, a user that wants to install new software on a Linux system, e.g., a more experienced Domain Owner that wants to install an Additional Service, typically requires privileged root access to the system. As soon as privileged users exist on the Domain Server, they will be able to abuse their rights to access or modify other Domain's data or tamper services. So either the usability of the Domain Server is low (if users do not have root permissions required to install new software) or the security will be impaired (if users have root access).

This dilemma can be resolved using *hardware virtualization*, see Section 9.1. Hardware virtualization is a technology able to provide so called *virtual machines (VM)* that can be assigned to the different stakeholders of the Domain Server. In each VM, an independent OS and any desired set of services can be installed and executed in almost perfect isolation of other VMs present on the same physical machine. The separation of stakeholders based on hardware virtualization is secure and also offers high flexibility as each stakeholder can obtain root access within her personal VM. It is for this reason that we have chosen hardware virtualization, more precisely, the Xen virtualization system, as basic building block for the architecture of the Domain Server.

10.3.2 Design of Stakeholder and Network Separation

In Figure 10.2 the details of the chosen Domain Server architecture based on the Xen virtualization system is shown. The figure primarily focuses on the isolation of different stakeholders (RA.1, RA.3, RA.5), the isolation of different service classes (RA.2) and on the networking requirements of the Guided Security Management System (RB.1, RB.2).

Overview

Each Domain's Security Services are contained in an own VM (domU), called *Security Service VM*, which isolates Security Services from other elements present on the Domain Server. Besides a Security Service VMs, one or several additional VMs (domU), which we denote as *Service VM*, can be assigned to a Domain for hosting Additional Services in an isolated environment. In the privileged dom0 no components exist that belong to a Domain of the network. Instead, dom0 will host the Auxiliary Services needed to operate the Domain Server, i.e., to allow users to setup and maintain their Domain's Security Services, see Chapter 11.

VMs for Security Services

Within each Security Service VM three (virtual) network interfaces are created. The purpose of these interfaces, eth0, eth1 and eth2, will be explained in the following.

The virtual network provided by Xen connects all eth2 interfaces located inside different Security Service VMs to a virtual interface vifx.2[1] located in dom0. All vifx.2 interfaces are connected by the network bridge br_reg in dom0 to the physical network interface wlan1 of the Domain Server. A *Host Access Point Daemon (hostapd)*, which *"is a user space daemon for access point and authentication servers"* [74], installed in dom0 spans the open Registration WLAN using the physical network interface wlan1. A DHCP server installed in dom0 is used to assign an IP address to each eth2 interface of Security Service VMs and to each physical device connected to the Registration WLAN in the IP range 192.168.0.0/24. This means, that the same Registration WLAN will be used by all Registration Services hosted on the Domain Server.

[1] vifx.y denotes interface y of guest VM x

The virtual eth1 interfaces of different Security Service VMs are internally connected to the vifx.1 interfaces of dom0. These are again bridged by br_serv to the physical wlan0 interface of the Domain Server. wlan0 is used by the hostapd installed in dom0 to span the EAP-TLS protected Service WLAN, i.e., the Service Network[2]. Again, IP addresses are assigned by the DHCP server installed in dom0 in the range 10.10.0.0/24 to virtual eth1 interfaces of VMs and to all physical devices connected to the Service Network. Therefore, devices of all Domains use the same Service Network.

The eth0 interfaces of Security Service VMs are connected to vifx.0 interfaces of dom0. These are again bridged to the physical network interface eth0 by br_ext. br_ext connects the Domain Server to the Internet. The Domain Server will use public IP addresses assigned by the network's Internet Service Provider (ISP). These public IP addresses are then used by the different Trust Exchange services inside Security Service VMs to publicly offer their services.

The need for several public IP addresses could currently by considered as a limitation as home networks or small company networks typically only obtain one public IPv4 address from their ISP. Nevertheless, this design was chosen with the advent of IPv6 in the public Internet in mind. Then home or small company networks will obtain a block of so called *IPv6 Aggregatable Global Unicast Addresses* identified by a 48-bit prefix, which is assigned by the ISP. The remaining 80 bit are at the free disposal for the network and can be used for the individual addressing of (virtual) machines and subnetworks.

[2]Please note: the architectural diagram does not show a physical interface for a wired Service Network. This, of cause, could be added easily but is omitted for the sake of brevity.

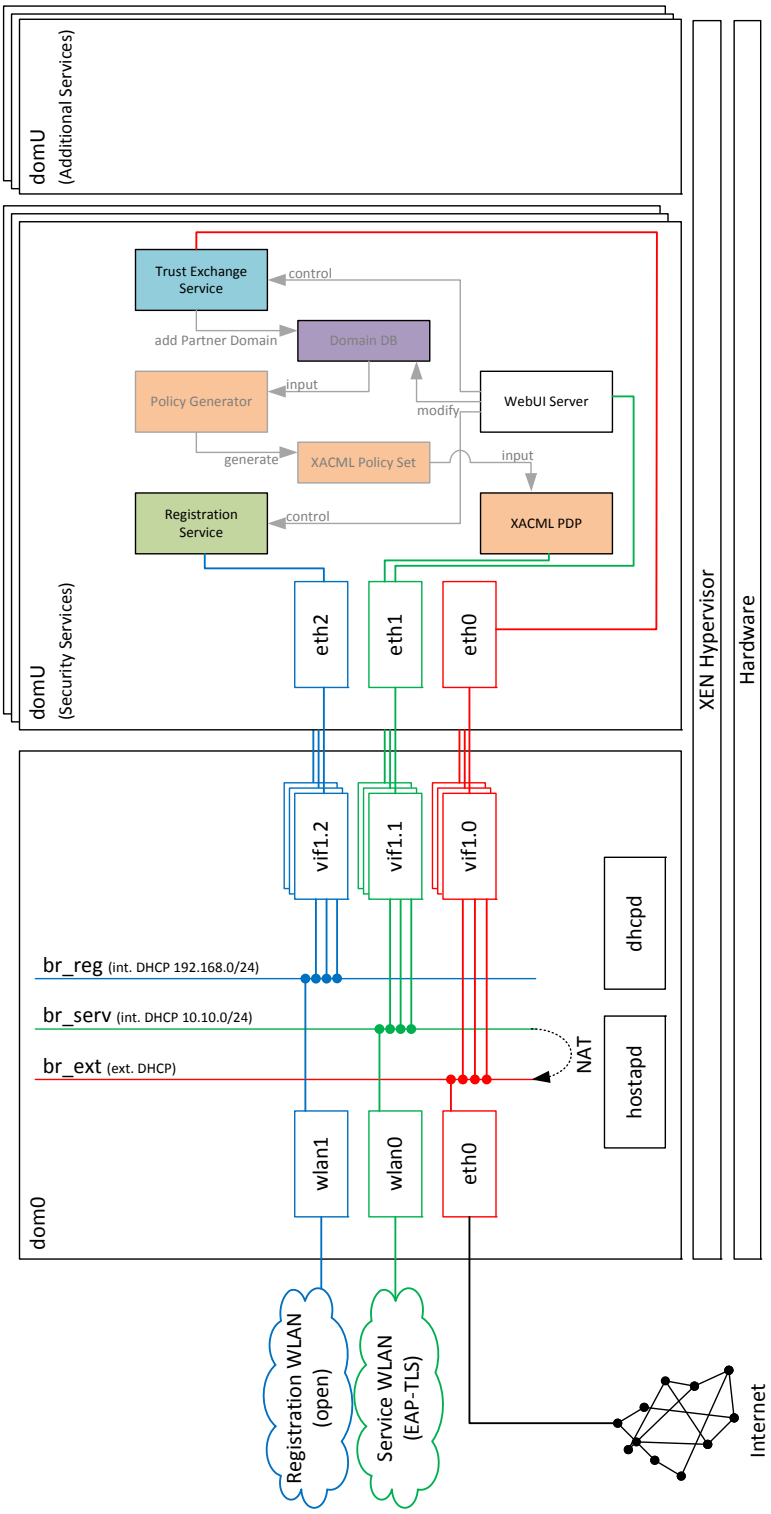


Figure 10.2: Domain Server architecture and virtual internal networking.

VMs for Additional Services

Up to now, the internal networking of Security Service VMs was described. For Service VMs, which for instance host media streaming or gateway services that allow access from the Internet to the inside of the local network (RB.2, RB.3), a quite equivalent internal virtual network setting is used, see Figure 10.3. The eth1 interface of a Service VM is internally connected to the Service Network and obtains an IP address from the DHCP server installed in dom0. This IP address is then used by local services installed inside the Service VM. The eth0 interface is connected to the public Internet and also obtains a public IP address from the ISP. These public IP addresses are then used by the OpenVPN Servers, which are installed in each Service VM. Additional interfaces, e.g., to connect to the Registration Network, are not necessary in Service VMs.

For remote access of devices, an OpenVPN service is running inside Service VMs. Mutual authentication of OpenVPN client (device) and OpenVPN server is performed using asymmetric keying material certified by trusted Domain CAs. In order to verify the certificates, OpenVPN client and server need to possess the Domain CA certificates that have signed the presented certificate. This can be achieved easily by registering own devices (see Section 5) and by exchanging trust with Partner Domains (see Section 6). After authentication, incoming network packets are routed from the virtual tun/tap interface [75] of the OpenVPN server in the Service VM to the Service Network.

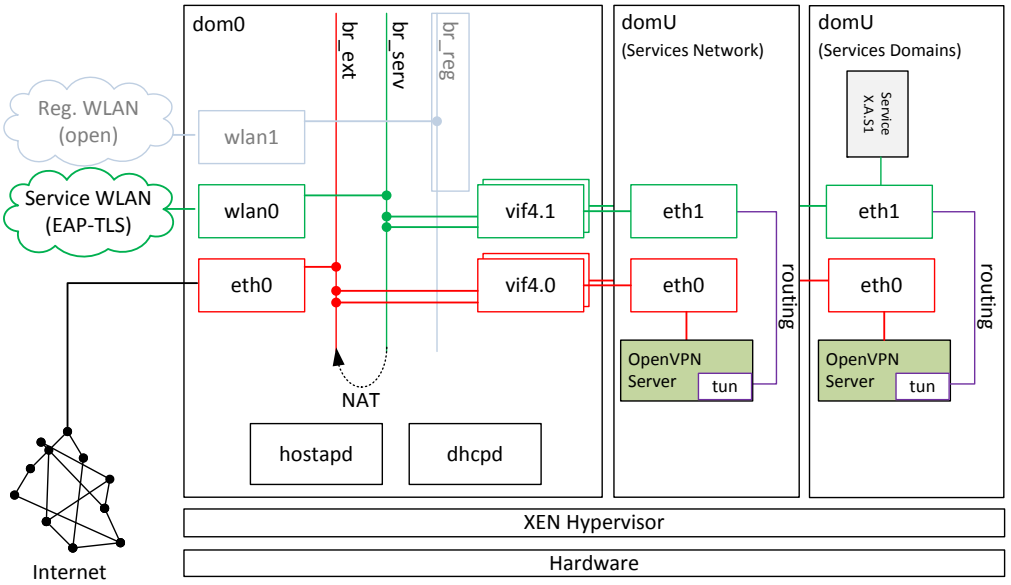


Figure 10.3: External Clients connect to OpenVPN endpoints of the Home Domain or User Domain

10.4 Discussion and Evaluation

10.4.1 Fulfillment of Requirements

The described compartmentalization concept based on hardware virtualization is an important step towards a secure and flexible multi-user Domain Server (RA). The required isolation/separation basically emerges from the fact that each Domain Owner (stakeholder) only obtains access to her personal VMs, which are used to host the Security Services of the Domain and Additional Services that belong to the Domain. However, VMs assigned to other Domains or the Domain Server's Auxiliary Services located in dom0 cannot be accessed, which has many beneficial effects.

First, a stakeholder cannot accidentally modify/tamper other VMs, i.e. violate the security of Security Services or services that belong to other Domains or the Auxiliary Services that control the Domain Server. This separation is still intact if a stakeholder possesses full (root) access rights within her own VMs. Therefore, the chosen solution offers full flexibility as stakeholders' rights do not have to be artificially restricted in order to protect the security and privacy of other stakeholders' VMs. Such a restriction would have been necessary on a Domain Server architecture not based on virtualization. Additionally, stakeholders that have full root access to a system might accidentally harm this system in case they are careless/inexperienced and apply faulty settings. E.g., a stakeholder might accidentally apply wrong network settings, which result in the inaccessibility of the whole system. Due to the usage of virtualization these negative effects will only have impact on one VM. Other Domains will not be affected, which is important to guarantee the availability of other Domains hosted on the Domain Server. Finally, malware, which might infest a VM after a user installed compromised software, can not spread easily between VMs. Negative effects caused by malware will therefore be contained within one VM. This is important to protect the security of other Domains.

Therefore, we are convinced that the requirements on secure and flexible separation of different stakeholders (RA.1, RA.3, RA.5) and between different service classes (RA.2) are fulfilled. Furthermore, we regard hardware virtualization as a good basis to address further requirements on the Domain Server in next chapters.

However, one can have doubts that the isolation between VMs is completely secure (RA.4). These doubts are reasonable as various examples exist, e.g., [76], that show that the isolation between Domains can be broken. In this paper security experts describe how they were able to escape from one VM, achieve access to the privileged dom0/the hypervisor, and could then compromise other VMs running on the physical machine. Nevertheless, the opinion of the same experts is that *"VMs can be much better isolated between each other than standard processes in monolithic kernels of popular OSes like Windows or Linux. This is because the interface between the VM and the hypervisor can be much simpler than in case of a traditional OS, and also the hypervisor itself can be much simpler."* [51, P. 5]. This means that virtualization will increase the security of a multi-user system by providing compartments for each stakeholder. But neither virtualization nor any other currently existing solution can guarantee that this isolation cannot be broken by highly sophisticated attackers or malware specifically designed to attack virtualization.

Furthermore, hypervisor security is a quite popular research topic at the time of writing this thesis. Quite practical approaches, such as presented in [77] or [51], aim to increase Hypervisor security by outsourcing exposed and easy to compromise elements, such as device drivers or the network stack, from the privileged dom0 to an unprivileged domU. In case these elements are compromised, the attacker only gets access to an unprivileged domU. This approach is highly useful and could be integrated into the Domain Server. Other research projects, e.g., [78], aim to increase the security of the Xen virtualization system using specific memory protection functionalities anchored in the hypervisor. Therefore, one can hope that solutions will be found that provide even higher security standards for hardware virtualization than exist today.

Besides the described beneficial security aspects, hardware virtualization contributes to fulfilling the requirements defined on the network topology (RB). The presented combination of virtual machines and virtual networking technology provided by the used virtualization solution Xen and the Domain Server's physical wireless and wired network interfaces provide the network topology required by Registration Service and other services (e.g. OpenVPN). The required isolation of networks is fulfilled as different, not connected, (virtual) networks are used to separate registration functionality and all other services. Therefore,

we are convinced that the Domain Server networking architecture fulfills the requirements RB.1 and RB.2.

10.4.2 Open Issues

A user that has root access to the privileged dom0 will be able to harm all VMs installed on the Domain Server. In the simplest case this user will be able to shutdown the VM assigned to a Domain, destroy this VM's image that contains all data of to the Domain, etc. More sophisticated attacks might be the unnoticed extraction of keying material of a Domain CA from the VM image and its later abuse, etc. Nevertheless, this problem can be resolved when no user has root access to dom0. In fact, the solution of requirement RC, the "Automation of Domain Management Tasks", will solve this problem. Once Domain management is automated, it is not necessary anymore that a user has root access to dom0. This approach is quite comparable to many Linux-based (embedded) products on the market, such as smart phones, WLAN routers, game consoles, etc., where users only interact with special interfaces with the device but do not have (and need) root access.

10.5 Conclusion

Most users of unmanaged networks have no deep experience in system administration. Therefore, they will not be able to setup and maintain the Guided Security Management System of their Domain on a self-maintained Linux server or to set up the network topology required by the Registration Service.

As an overall solution to this problem we proposed a novel device, called Domain Server, that hosts the Guided Security Management Systems of different Domains, enforces the required network infrastructure and also provides Auxiliary Services that enable the inexperienced users to setup and maintain their personal Domain's Guided Security Management System.

The central aim of this chapter was to create the basic architecture of this device. The Domain Server architecture is mostly defined by the need for the described secure and flexible multi-user/multi-Domain hosting and by finding a concept to provide a suitable and secure network topology for the Security Services and other services running on the Domain Server.

After identifying different stakeholders and requirements on the network topology, we proposed the design of the Domain Server architecture. The Domain Server architecture leverages hardware virtualization to separate the different stakeholders and a sophisticated combination of virtual and physical networking to provide and separate the required network topology.

On the basis of the presented Domain Server architecture, further Auxiliary Services will be presented in subsequent chapters that finally enable the inexperienced users to effortlessly manage their Domains and to take care for their Domain's security and resilience.

Key Findings and Contributions

▷	The Guided Security Management System introduced previously has various security requirements on the environment it is hosted on and requires a specific network topology. Inexperienced users will not be able to setup and maintain this environment or network topology.
▷	A ready to use device, the Domain Server, is needed to host the Guided Security Management System and offer a turnkey experience to inexperienced users.
C10.1	**Domain Server architecture** based on the XEN hardware virtualization system.

11. Auxiliary Services for Automated Domain Management and Resilience

In the previous chapter we presented the architecture of the Domain Server based on hardware virtualization. The Domain Server provides the required network topology and serves as a secure and flexible host for Security and Additional Services of the Domains that belong to the local network. The mere architecture does not yet answer the question how inexperienced users can setup and maintain their Domains on their own or take care for their resilience.

In this chapter these issues, i.e., the requirements "Automation of Domain Management Tasks" (RC) and "Availability and Resilience of Domains" (RD) defined in Section 8.2, will be addressed.

Chapter Structure

After the refined problem analysis performed in Section 11.1, requirements on the solution are defined (see Section 11.2). Subsequently, the design of two Auxiliary Services that solve the addressed problems will be presented in Section 11.3 and discussed in Section 11.4. The chapter is concluded in Section 11.5.

Acknowledgments

This chapter contains results developed by Simon Mittelberger in his Master's Thesis [38] and accompanying work as student research assistant. These works were assigned and supervised by the Author in the context of his doctorate.

11.1 Refined Analysis

First of all, a detailed analysis of the problems addressed in this chapter is performed. This analysis will provide a deeper understanding of the addressed issues and will help to define requirements of the mechanisms that are suitable to resolve them.

Automation of Domain Management Tasks

As explained before, the mostly inexperienced users present in unmanaged networks do not have the ability to setup and maintain the Security Services of their Domain and

to maintain the device that hosts these services. Furthermore, the concept of hardware virtualization is quite uncommon for most ordinary users and must be regarded as an additional obstacle. Therefore, users need assistance in order to perform tasks related to the management of the Domain Server and Domains on their own. A way to resolve these issues is to apply the automated/guided management paradigm again.

In the following the management of the Domain Server with regard to the Security Service VMs assigned to Domains is addressed in detail, the management of VMs used for Additional Services briefly. Basically three different management tasks exist in this context that need to be supported by Auxiliary Services.

The first task the Auxiliary Services of the Domain Server must support is the *creation* of a new Domain. This task needs to be performed, for instance, when a new home is set up (create Home Domain) or if a new user should be added to an existing local network (create User Domain). For creating a new Domain in the local network, a new Security Service VM needs to be created, the Security Services must be installed therein and finally a new Domain CA key must be generated. As these tasks are too complicated to be performed by most users, they must be completely automated by the Auxiliary Services of the Domain Server.

As a Domain is a virtual construct that is mostly independent of the rest of the local network it is possible to move a Domain to another network. This management task is needed if the Domain Owner moves with her devices to a new home, for instance. To implement this procedure the Security Service VM of this Domain must be *migrated* to another Domain Server. The migration of Domains is the second management task that needs to be automated by Auxiliary Services.

Finally, the need exists to *destroy* a Domain when a Domain Owner does not want to use the system anymore, for instance. As the VM image contains critical information it must be safely destroyed by the Auxiliary Service as otherwise sensitive information, such as the key of a Domain CA, might be abused. The secure destruction of a Domain is the third task that must be automated by Auxiliary Services.

Availability and Resilience of Domains

A Domain is defined by a set of information. This information is generated by the Guided Security Management System at the point in time when the Domain is created, e.g., the Domain CA's private key, or collected and created over time, e.g., during the Trust Exchange process with another Domain or when the Domain Owner configures access control settings. The whole of information is called the *state* of a Domain.

Losing the state of a Domain will have negative impacts on the availability of services that belong to this Domain and will cause major inconveniences to the Domain Owner. Therefore, the resilience of a Domain's state is crucial for the overall *resilience* of this Domain and, in the end, for the availability of the Domain's services. The term *Domain resilience* denotes the ability of a Domain to provide and maintain an acceptable level of service in the face of various faults[1].

The goal of the following analysis is to understand how the resilience of a Domain can be guaranteed in order to achieve high availability of all services that belong to this Domain. The first step of this analysis is to understand which different state items exist.

The private key of the Domain's CA is the first and maybe most crucial information that belongs to a Domain's state. If this key is lost no new devices can be registered

[1]Definition derived from the definition of *network resilience* given by Sterbenz et. al. [79].

unless a new private key is created for the Domain CA. Creating a new Domain CA must be accompanied by reregistering all devices that belong to the Domain, i.e., all certificates must be recreated and signed by the new Domain CA. Additionally, all old Trust Relationships need to be reestablished, as the new Domain CA's signing key is untrusted (unknown) by the Partner Domains.

The next state item is the Domain's database that contains information collected by the Trust Exchange Services. The database stores all received certificates of trusted partner Domains and additional meta-information such as owner names, Identification and Reputation Levels, etc. If this information is lost, the Trust Relationships are destroyed and need to be reestablished. The database additionally contains the access control settings needed to generate the XACML Policy Set of the Domain. If these settings are lost, the Domain Owner must recreate them.

The previous considerations have shown that reestablishing lost state is cumbersome for the Domain Owner. But besides this described inconvenience, lost state will also negatively affect the accessibility/availability of services that belong to the Domain. Services will become unavailable, for instance, when no XACML Policy Set can be generated, as the Policy Decision Point is unable to decide if access to a service should be granted and therefore permits access per default Hence, the resilience of a Domain's state and the ability to quickly restore it in case state is lost are highly important steps to guarantee the availability of a Domain's services.

Two other issues that must be considered in this context are, first, that a Domain's state changes continuously. Therefore, it is not sufficient to create a backup of state only now and then. Otherwise partial loss of state must be expected. Instead, the service that protects the resilience of a Domain must backup the state whenever it changes.

Second, an additional difficulty in unmanaged networks is that typically no highly available and secure storage/backup server exists here that might receive the continuous state backups of Domains. A possible solution might be to use a network-based storage service provided outside the local network, e.g., a cloud-based storage service located in the Internet. Storage services outside the local network might cause security and privacy issues if no suitable precautions are taken. Therefore the state backups must be protected using state of the art cryptography that prevents modification and protects the confidentiality of this information.

Again, it cannot be expected that inexperienced users will continuously backup their Domain's state to a reliable network-based storage solution. Therefore, an automated, easy to use and transparent service is needed that takes care for the resilience of Domain state.

11.2 Requirements

Based on above analysis the following requirements are defined on the Auxiliary Services hosted on the Domain Server.

Automation of Domain Management Tasks

- **RC.1: Support of Creation, Destruction and Migration of Security Service VMs:** The *Domain Server Manager* of the Domain Server must support the creation and destruction of Security Service VMs as well as moving Domains to another Domain Server.
- **RC.2: Ease of use:** Users must be able to perform the named management tasks without having expertise in system administration, i.e., the Domain Server Manager must be automated, transparent to the user and easy to use.

Availability and Resilience of Domains

- **RD.1: Automated and Continuous Resilience:** The *Resilience Service* must work automatically and be transparent to the user.
- **RD.2: Quick Recovery of Lost State:** In case a Domain's state is lost, the saved state needs to be quickly restored by the Resilience Service in order to prevent long downtimes of services, etc.
- **RD.3: Security of State Backups:** The confidentiality of the Domain's state backups as well as their integrity and authenticity must be ensured.

11.3 Design

In this section the design of the Domain Server Manager and Resilience Service is presented, as both services work cooperatively.

11.3.1 Storage of Backups

As explained before, the Resilience Service needs some highly available target where the state backups can be stored continuously and permanently. It might be possible to use some secondary storage medium, a USB hard drive, for example. This would be sufficient for most situations, e.g., a hardware defect destroys all state on the Domain Server. But this solution would not cover problems caused by, e.g., a house fire that destroys the Domain Server and the attached USB drive. Moreover, the solution would be inflexible in case the state of a Domain needs to be moved to another Domain Server.

Therefore, a networked storage solution is required that stores the state externally of the local network and that can be accessed from everywhere. A system as Tahoe, which is *"a distributed file system, which safely stores files on multiple machines to protect against hardware failures"* [80] might be suitable. The file system could either be distributed on machines within the same network or for additional robustness against catastrophes be distributed among the Domain Servers of various networks. Alternatively, cloud-based storage and backup solutions already widely used, such as Dropbox [81], Spideroak [82], etc. might be utilized. However, storing state on machines or services external the own Security Service VM is problematic from a security point of view. Therefore, the integrity and authenticity of state backups must be secured using well-known, state of the art cryptographic encryption mechanisms to protect the data's integrity and confidentiality.

11.3.2 Templates for Virtual Machines

The state of a Domain is stored within the VM image of a Security Service VM. A naive approach for establishing resilience of a Domain might be based on backing up the whole VM image. This is neither necessary nor beneficial. An important insight is that the only difference between Security Service VMs is the state contained within their images and some other, easy to recreate settings, e.g., considering the network. The operation system and all Security Services are the same. For this reason, it is sufficient that the Resilience Service only protects the state of a Domain contained inside the VM image. The rest of the Domain's Security Service VM is completely disposable.

This property is crucial as it allows designing the required mechanisms based on so called *Security Service VM templates*. As a VM template, a VM image is denoted that contains a ready to use operation system with preinstalled, ready to use Security Services. The Security Service VM template does not contain Domain state or any additional configurations. VM templating simplifies the design of the required Resilience Service and of the Domain Manager Service much.

The creation of a new Security Service VM for a new Domain becomes as simple as creating a copy of the Security Service VM template. Additional minor modifications are subsequently needed, such as generating a new private key for the Domain CA and applying network settings. Finally the just created Security Service VM can be started.

During normal operation, the Resilience Service backups the Domain state to a suitable storage medium. In case a Security Service VM was destroyed by accident, the Domain can be recovered easily as the Domain's state was saved and can be simply installed into another Security Service VM created from the template.

Migrating an already existing Domain is basically a combination of destroying and recovering a Domain. At first, the Security Service VM image is destroyed on the "source" Domain Server. Then, a new VM image is created from the Security Service VM template on the "destination" Domain Server and finally the saved Domain state is recovered there.

11.3.3 Interaction of Domain Owner and Domain Server

In order to trigger the tasks related to Domain management the Domain Owner must somehow interact with the Domain Server Manager.

For the interaction with components of the Guided Security Management System we offered web-based graphical user interfaces. Such WebUIs might be used in this context as well. However, we argue that another, much simpler, solution is better suited. We propose to utilize cryptographic token (see Section 9.2.1) assigned to a Domain Owner for a very simplistic and natural way of interaction with the Domain Server.

The overall idea is to offer an easy to comprehend, key-like metaphor to the user. Basically speaking, if the cryptographic token of a Domain Owner is plugged into the Domain Server, the Security Service VM of the Domain assigned to the token is started. If the token is removed at a later point in time, the VM is stopped and destroyed.

A further benefit of this solution is, that the cryptographic key contained inside the token can be used by the Resilience Service to encrypt the backups of the Domain state. As asymmetric cryptography is slow, a hybrid encryption scheme must be used. A symmetric key, the *backup key*, is used to encrypt the backup. The backup key in turn is encrypted with the public part of the asymmetric key contained on the token. Both, the encrypted backup and the encrypted backup key can be stored securely on some networked storage service.

11.3.4 Services for Domain Maintenance and Resilience

Our solution for automated Domain management and resilience is depicted in Figure 11.1. The user only needs to interact with the *Domain Server Manager* using her personal cryptographic token. The rest is performed automatically and transparently to the user.

11.3.4.1 Workflow of the Domain Server Manager

The left hand side of this figure displays an activity diagram of the Domain Server Manager. All tasks of this component are related to the management of Xen VMs. Therefore, the Domain Server Manager must be located in the privileged dom0, which is also beneficial for the security of this component, as it is isolated from the rest of the Domains.

After starting the Domain Server, the Domain Server Manager is started as background service (Step 1). Subsequently, the manager tests if there is a sufficient amount of spare disk images available (Step 2), i.e., a copy of the Security Service VM template. If not, a new copy is created (Step 3). The manager always holds a certain amount of spare disk

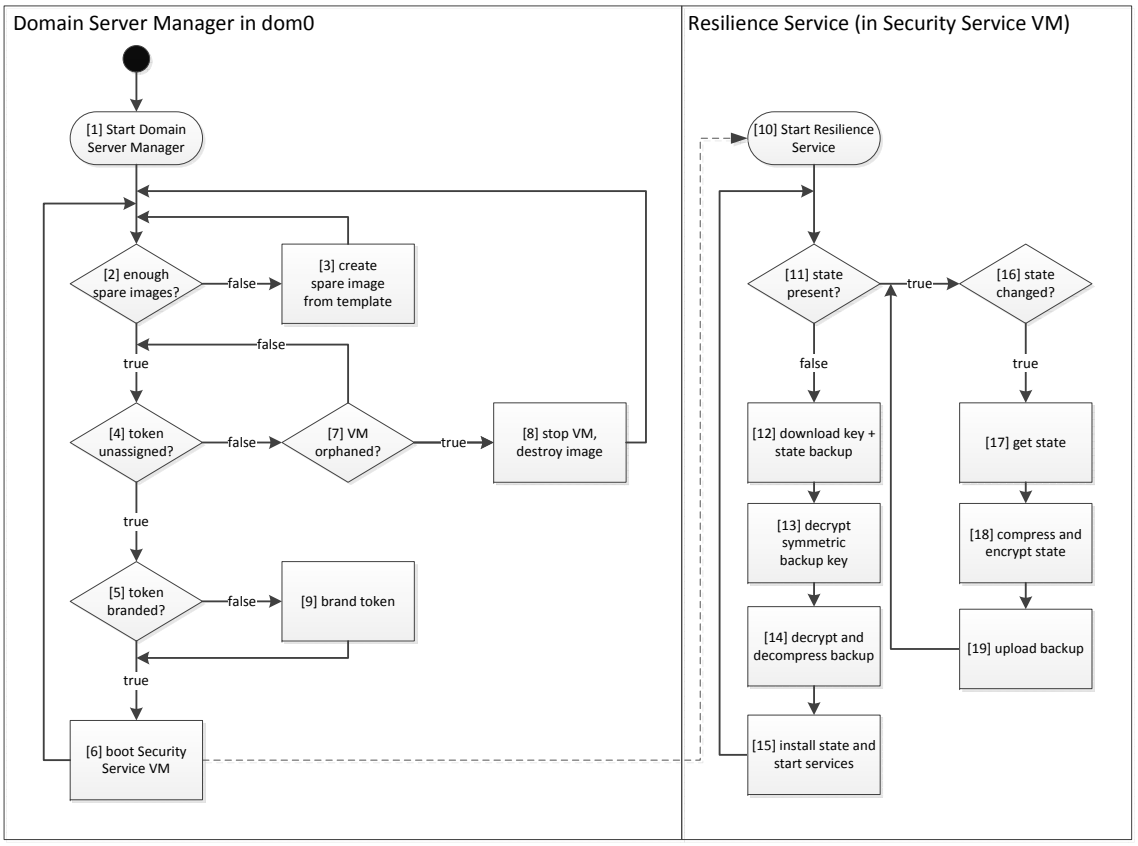


Figure 11.1: Interaction of Domain Server Manager and Resilience Service

images to guarantee short response times for the case that a Security Service VM needs to be started.

Subsequently, the manager waits for cryptographic token to be plugged into the Domain Server. If the manager detects the presence of a token (Step 4), the manager tests if this token is already *branded* (Step 5). In this context "branding" means, that an asymmetric key, which is used later by the Resilience Service to encrypt/decrypt the backup key, is available on the token.

In case the token is already branded, an empty Security Service VM is launched (Step 6) using a previously generated spare disk image. After the Security Service VM is started, the Resilience Service running inside the just started Security Service VM restores the Domain's state, for details see Section 11.3.4.2. In case an inserted token is unbranded, the Domain Server Manager advises the token to generate a new asymmetric key (Step 9). As before, a new Security Service VM is booted. But, as the Domain is new, no state backup is available that must to be restored by the Resilience Service in this VM later.

Another important task of the Domain Server Manager is shutting down VMs and destroying their VM images. This action is triggered when the manager detects so called *orphaned* VMs (Step 7). Orphaned VMs are running Security Service VMs whose assigned token was removed. If an orphaned VM is found, the VM will be stopped and its VM image destroyed/overwritten (Step 8).

11.3.4.2 Workflow of the Resilience Service

The activity diagram of the Resilience Service is depicted on the right hand side of Figure 11.1. The Resilience Service runs inside a Security Service VM and is started automatically (Step 10) as soon as the VM has finished its boot process.

When the Resilience Service detects that no state is present (Step 11) it begins with state recovery. To recover state, the symmetric key encrypted with the public key of the user's token and the encrypted backup needs to be downloaded from the used network storage first (Step 12). The symmetric backup key is then decrypted using the token's private key (Step 13). Finally, the backup key is used to decrypt the state backup (Step 14). The state items contained in the backup are then installed and the Guided Security Management Service and all related services started (Step 15). After recovery, the Resilience Service checks the different state items periodically for changes (Step 16). When a change is detected, the Resilience Service collects all state items (Step 17), encrypts this information using the symmetric backup key (Step 18) and uploads the backup on the networked storage service (Step 19).

11.4 Discussion and Evaluation

The described solutions for Domain management and resilience satisfy all requirements we have defined in Section 11.2. Furthermore, we are convinced that the combination of both services based on the proposed VM templating is quite elegant and highly satisfactory for users. Additionally, the solution provides high flexibility and is a good basis for further mechanisms targeted to the integrity of execution environments, which are explored in the subsequent chapter.

11.4.1 Fulfillment of Requirements

The proposed solution satisfies all requirements on usability, as the user does not need to have any understanding of system administration to create, destruct or migrate the Security Service VM assigned to her Domain. The reason for this is that the Domain Manager (RC.1, RC.2) performs all necessary tasks automatically after the user triggered this service by plugging or unplugging her personal cryptographic token into or from the Domain Server. Depending on the context, plugging the token into the Domain Server will either trigger the creation of a new Domain, recovery of an already existing Domain or migration of an existing Domain to another Domain Server. Unplugging the token from the server will trigger the destruction of the Security Service VM image assigned to the token.

As specified, the Resilience Service is completely transparent to the user, i.e., works automatically and without any interaction needed between user and service. Depending on the context, the Resilience Service either saves state changes automatically to a network storage service (RD.1) or restores Domain state from this networked storage (recover paused or destroyed Domain; RD.2).

The Resilience Service cannot guarantee the permanent availability of the Domain's services but guarantees the resilience of a Domain for the case of hardware or software failures or the destruction of the Domain Server hardware that leads to loss of state. The latest Domain state can always be recovered from the network storage.

For the user the proposed way of interacting with the Domain Server Manager using a cryptographic token is highly intuitive, easy to perform and convenient (RC.2). This is, first, because a key-like metaphor is used that can be understood easily and, second, the token satisfies the need of the Resilience Service for a cryptographic key that can be used to encrypt the state backups (RD.3). Without the token, the user would have needed to trigger actions using a GUI and provide the cryptographic key manually. This would not be as convenient for the user as the proposed solution.

11.4.2 Resilience of Cryptographic Token

Besides the positive aspects of the used cryptographic token, drawbacks emerge from these token that must be described as well.

First, the token plugged into the Domain Server might be stolen from an adversary that has physical access to the Domain Server. As a result, the Security Service VM will be unavailable as removing the token will shut down the Security Service VM. Additionally, the Domain cannot be recovered anymore by its legitimate owner, as the key contained in the token is needed to decrypt the Domain state. Furthermore, the adversary might use the token to access the stolen Domain's state, e.g., to abuse existing Trust Relationships to other Domains. These problems can be mitigated when the introduced key metaphor is softened. Instead of requiring the token to be plugged into the Domain Server all the time, a modified solution could be created where it is sufficient to plug in the token only for a while and remove it afterwards.

Additionally, it is important to understand that the token is a single point of failure for a Domain's resilience. In case the token is defect or destroyed the Domain cannot be recovered, as the private key contained in the token is lost. This problem can be mitigated quite easily. The key must be generated in software on a secure computer system and imported into the token afterwards. Additionally, a backup of this key must be kept in a safe place, e.g., a bank deposit, etc.

11.4.3 Benefits of VM Templating

Instead of creating Security Service VMs from scratch by installing OS and applications in an empty VM image, which would be quite inefficient and complicated to automate, it is easier and more efficient to provide ready to use VM images that are used as templates for Security Service VMs. Such ready to use VM images are also called *virtual appliances*.

Besides this simplification, VM templating offers additional benefits, such as a simple way of updating VMs. In case a component running inside the Security Service VM must be updated or the VM's operation system must be updated or upgraded, the Domain Server Manager simply needs to download the latest version of this virtual appliance at some point in time and use the new image from now on to spawn Security Service VMs.

The question remains from which resource the Security Service virtual appliances can be obtained. The answer differs. If we regard the Domain Server as a product, the manufacturer of the Domain Server will provide the virtual appliances as today's manufacturers of devices provide firmware images for their devices. If we regard the Domain Server as a highly customized Linux distribution, the corresponding Linux/open source community might provide the virtual appliance.

The overall benefit of this approach is that virtual appliances are well maintained and tested from professionals or highly experienced users before they are rolled out. To some extent the effort that is spared within the local network to setup and maintain Security Service VMs is shifted to the outside of the home.

11.4.4 Performance

We implemented a prototype of the Domain Server and conducted some simple performance tests, which we think are completely satisfactory. The prototypes are based on the Ubuntu 12.04 LTS operation system and the Xen 4.1 hypervisor. The used hardware is a mini-PC equipped with a dual-core 2 GHz Intel i5 processor, 4GB of RAM and a standard 2.5 inch hard drive.

The Domain Server always keeps empty spare images. So when a Security Service VM needs to be started, no copy of the template image must be created first. For this reason a Security Service VM is ready on our Domain Server prototype after less than two minutes after a user plugged in her cryptographic token.

Recovering a Domain consumes an insignificant amount of time, several seconds, more. The extra time is needed to download the Domain state from the network storage mechanism and to decrypt and install the information at the right place inside the just started Security Service VM.

Depending on Domain size, age and number of established Trust Relationships to other Domains, the size of the backup will grow. Nevertheless we do not expected that the backup will become large enough to slow down recovery significantly. For this reason, also slow Internet connections will be unproblematic.

11.5 Conclusion

The previous chapter defined the basic architecture of the Domain Server but did not answer the question how inexperienced users can be enabled to setup and maintain the Guided Security Management System within their own VM. An additional issue is the resilience of Domains, which mostly depends on a Domain's state (Domain CA key, access control settings, etc.) This information needs to be backed up to a safe location permanently. As a solution to both problems we proposed a concept based on the virtualized Domain Server that incorporates an automated Domain Management and Resilience Service.

The Domain Server Manager uses prefabricated virtual appliances for Security Service VMs that contain operation system, the Guided Security Management System and the Resilience Service. The Resilience Service performs backups of the valuable Domain state automatically or restores the state when needed. The backups are encrypted so that it is possible to use storage services external the home, e.g., in the Cloud.

Another question we answered is how the user can interact with these services. We proposed the usage of a cryptographic token which 1) triggers the described services and 2) serves as a secure container for the encryption key needed by the Resilience Service. Starting a Domain's services becomes as easy as plugging the card into the Domain Server with this solution.

The presented, incorporated solution for Domain management and resilience resolves the addressed issues, is quite efficient and elegant. The presented technologies are finally a good basis for addressing issues related to the integrity of Security Service VMs, as we will discuss in the subsequent chapter.

Key Findings and Contributions

▷	Inexperienced users are not able to setup the Guided Security Management System of their Domain on the Domain Server or are able to take care for the Domain's resilience on their own.
▷	Based on the previously introduced Domain Server architecture, suitable concepts and services that automate setup and resilience of Domains are needed.
C11.1	The **Domain Server Manager** automates the processes needed to setup a Security Service VM that hosts a Domain's Guided Security Management System.
C11.2	The **Resilience Service** performs backups of the valuable Domains state or recovers state when needed automatically.

12. Trust and Integrity of Domain Server and Virtual Machines

Until now we have presented the basic architecture and various functionalities of the Domain Server. The Domain Server acts as host for virtual machines assigned to different Domains that belong to the local network, enforces the network topology required by the Guided Security Management System and finally offers Auxiliary Services that enable Domain Owners to manage the VMs assigned to their Domains effortlessly.

One important aspect that remained unaddressed so far is how the "Trust and Integrity of Execution Environments", i.e., the integrity of the Domain Server itself and of Security and Service VMs, can be guaranteed. This is one of the overall requirements (RE) of our architecture, which will be addressed in the present chapter.

Chapter Structure

First, a refined problem analysis is presented in Section 12.1, which leads to the definition of requirements on the solution (see Section 12.2). After discussing the state of the art in Section 12.3, the problem's solution that consists of several collaborating mechanisms is presented in Section 12.4. In Section 12.5 a further possible application of the developed technology is explained. The proposed technologies will be discussed and evaluated in Section 12.6. Finally, the chapter is concluded in Section 12.7.

Acknowledgments

This chapter contains results developed in the Bachelor's Theses of Martin Erich Jobst [83] and Michael Dorner [84]. Both works were assigned and supervised by the Author in the context of his doctorate.

12.1 Refined Analysis

The Guided Security Management System is the central authority of a Domain. Valid identities are distributed to devices and services, such as media streaming, data storage, control of home or building appliances, the local wireless LAN, etc. Furthermore, the Policy Decision Point hosted inside the Security Service VMs also authorizes access requests of devices to services that belong to this Domain. If a Security Service VM is compromised, access control to all services can be broken. For this reason the Domain Server itself and the Security Service VMs are high value targets for adversaries that attack our system.

For the active protection of computer systems or VMs, numerous soft- and hardware-based approaches exist already. Examples for software-based approaches include Host Intrusion Detection Systems (HIDS) or Host Intrusion Prevention Systems (HIPS), virus scanners, rootkit detectors, etc. HIDS, for instance, actively search for anomalies on a computer system in log files, process lists, (configuration) files, etc. in order to find evidence that the system is under attack or has already been compromised. If such problems are detected, HIDS typically send reports to the system administrator who must then address the reported problem. HIPS additionally have the ability to address some detected attacks or threats actively by invoking suitable countermeasures. For instance, HIPS are able to block the attacker's IP address, disable a compromised user account or terminate unknown applications.

We argue, that the combination of available protection mechanisms and the isolation between VMs assigned to Domains and dom0 where Auxiliary Services reside is sufficient to protect the Domain Server and individual VMs against attacks or malware that spreads in the local network or the Internet. HIDS/HIPS and related technologies prevent that single VMs can be compromised. If HIDS/HIPS fail, isolation between VMs prevents that an successful attacker is able to compromise other VMs or the complete Domain Server. For this reason, the just mentioned protective mechanisms need to be integrated into all VMs running on the Domain Server.

Authenticity and Integrity of Virtual Appliances is Crucial

A threat worse than attacks on individual running Security Service or Service VMs emerges from the collectively used virtual appliances. In Section 11.4 we already described that the virtual appliances for Security Service VMs might be prefabricated and distributed by an authority we refer to as the *image maintainer*. The maintainer might be, for instance, the manufacturer of the Domain Server or the open source/Linux community. Comparable to virtual appliances that contain the Guided Security Management System of a Domain, any service or set of services might be packed in and distributed as virtual appliance, e.g. media streaming, file storage or home control services.

An adversary might abuse this and modify authentic virtual appliances and distribute bogus versions using a compromised download mirror, for instance. Bogus virtual appliances might contain tampered components of the Guided Security Management System, which might nullify the effectiveness of our system. Alternatively, any type of malware might be distributed inside a bogus virtual appliance.

It is quite obvious that an attacker that successfully distributes such manipulated virtual appliances will harm numerous Domains. Thus we regard the distribution of bogus virtual appliances as a rewarding attack vector that must be closed.

Integrity Verification of Virtual Appliances vs. Domain Server Integrity

Comparable to code signing concepts for applications, such as proposed by Apreville [85] or IBM [86], virtual appliances should be signed by their maintainer. This gives the opportunity to validate the authenticity and integrity of a virtual appliance by the consumer of this image before it is used. The functionality needed to verify images of virtual appliances can be included into the Domain Server or, more precisely, into the Domain Server Manager, which was already described in Section 11.3.4.1.

However, one must consider the option that the Domain Server Manager might not have *integrity* itself, as this service or another program executed on the Domain Server was compromised by an successful attacker or even by the malicious owner of the Domain Server. Therefore, the Domain Server cannot be *trusted* to correctly verify the integrity

of a virtual appliance before it is used. For this reason the user of a VM cannot be sure that its authenticity and integrity have been verified correctly. Therefore, a mechanism is needed that is able to prove the Domain Server's integrity to the Domain Owner. For a definition of the terms integrity and trust refer to Section 9.3.1.

Please note: Besides the just described integrity verification of virtual appliances, the Domain Server Manager performs various other tasks, which are highly critical for the security of a Domain. For instance, the Domain Owner must be sure that the Domain Server Manager really destroys the VM image of her Security Service VM when she unplugs her smart card from the Domain Server, etc.

Confidentiality for Persistent Service VM Images

Besides the already discussed problems that emerge from bogus virtual appliances another problem exists. In contrast to Security Service VMs, which are destroyed/disposed when the Domain Owner unplugs her smart card, VMs might be customized by more experienced users. The image of such a VM needs to be stored *persistently* on the Domain Server. The problem is that VM images are naturally without any protection, so various ways to harm the image and its user exist. For instance, when a VM is not in use an attacker that controls dom0 of the Domain Server might mount the image and extract valuable information from it. Alternatively she might replace applications with bogus versions in the same way. The same kind of attacks must be feared when the VM image is transferred via an unprotected network from one Domain Server to the other.

By these reasons, VM images stored persistently on the Domain Server must be protected against espionage of data contained in the image and against modification.

12.2 Requirements

Based on the analysis performed above, the following requirements are defined for a solution suitable to guarantee the trustworthiness and integrity of Domain Server and VMs assigned to different Domains:

- **RE.1: Proof of Domain Server Integrity to the Domain Owner:** The Domain Owner needs to trust the Domain Server, in particular the verification and protection mechanisms of virtual appliances. Therefore, the Domain Server's integrity must be guaranteed and proven to the Domain Owner.
- **RE.2: Authenticity and Integrity of Virtual Appliances:** The virtual appliance of a Security Service or Service VM must be signed by its maintainer in order to protect it against unauthorized modifications and to prove its origin and authenticity.
- **RE.3: Verification of Virtual Appliances:** The Domain Server Manager that creates/starts VMs from virtual appliances must verify the authenticity and integrity of the image before it is used to start a VM.
- **RE.4: Integrity Protection and Confidentiality of Persistent VMs:** Customized VM images stored persistently on the Domain Server must be encrypted in order to prevent against espionage of data contained inside the image and unauthorized modification.
- **RE.5: Decryption of Private Data after Successful Integrity Verification only:** Private information that belongs to a Domain may be decrypted on the Domain Server or on a VM running on the Domain Server only, after the integrity of the Domain Server and VM was successfully verified.

- **RE.6: Ease of Use:** The mechanisms needed to address the above stated requirements may not diminish the usability and user friendliness of the system.

12.3 Discussion of the State of the Art

The state of the art describes various mechanisms suitable for integrity verification of computer systems, which allow users/other parties to establish trust into this system. In the following their usability in our scenario is assessed.

12.3.1 Secure Boot Concepts

A concept able to verify the integrity of a computer system and to implicitly prove its successful verification to a user is *Secure Boot*. Secure Boot can be described as a mechanism that only allows a computer to finish its boot process if the integrity of already loaded parts of the system could be verified. When a computer system equipped with Secure Boot has successfully finished its boot process, a user implicitly knows that the verified system components have integrity and are trustworthy.

Two implementations of the Secure Boot paradigm exist, which are explained briefly and assessed in the following.

UEFI Secure Boot:

The Secure Boot concept standardized by the UEFI forum (Unified Extensible Firmware Interface) is software-based [87]. Here the major idea is to actively verify signatures of system components (boot loader, kernel, etc.) and to load the component only if its authenticity and integrity could be verified. Otherwise, the system start is aborted or the system boots into a maintenance mode.

TCG Secure Boot:

Instead of performing active verification of system components, the Trusted Computing Group (TCG) follows a different approach, which is based on their sealing and integrity measurement concepts. The following description requires basic knowledge about Trusted Computing mechanisms, which were introduced in Section 9.3.2.

Sealing will only allow to use the private part of a specific asymmetric key contained in the TPM, if the TPM's PCRs contain values that were predefined at the point in time this key was created. TCG denotes this class of keys as *Sealing Keys*. The predefined PCR values are the fingerprints of expected and trustworthy system components. After its creation, the public part of the Sealing Key is used to encrypt a symmetric key, which in turn encrypts all components of the computer system loaded after the boot loader, e.g., operation system kernel, modules and applications.

During the early stages of the Trusted Boot process fingerprints of BIOS and the Trusted Boot Loader are created and written into PCRs. If the operation system kernel should be loaded by the Trusted Boot Loader, this is only possible if the TPM allows to use the private part of the Sealing Key. This is the case, if fingerprints of the previously loaded BIOS and Trusted Boot Loader are as predefined.

Assessment

TCG Secure Boot is inflexible, as exactly one system configuration can be expressed by the PCRs. UEFI Secure Boot is more flexible as software signatures are validated. This leaves some space for different valid system configurations. However, UEFI Secure Boot is based on software, which is comparably easy to manipulate even by software-based attacks, whereas manipulating TPM hardware and the CRTM code deeply integrated into the system requires physical access to the device.

Furthermore, both outlined Secure Boot concepts have various drawbacks. First, a system that has finished its boot process might have passed the integrity verification of Secure Boot. But it might also be possible that this computer does not use Secure Boot at all. For instance, an adversary with access to the hardware, e.g., the owner of the Domain Server, might have manipulated the computer system and disabled Secure Boot. Thus, a user cannot be sure that the machine in question did indeed complete the integrity verification as intended.

The second drawback is that current Secure Boot concepts focus on system components loaded at early boot phases, such as the BIOS and the boot loader. The integrity of applications loaded after the actual boot sequence is finished, e.g., the Domain Server Manager, is typically not guaranteed.

12.3.2 Integrity Measurement and Remote Attestation

The Integrity Measurement and Remote Attestation concepts were already described in Section 9.3.2.4. At this place only a short summary is given to refresh the reader's memory.

The core idea of Remote Attestation is to transport a sequence of fingerprints of software components already loaded and executed on one computer system to another computer system, which is called *verifier*. Creation and transport of fingerprints are done in a trustworthy and verifiable manner supported by the TPM of the system. The verifier then is able to assess the reported fingerprints and finally establish trust into the integrity of this system.

Assessment

The Trusted Boot concept and its extensions (TrustedGRUB, IBM IMA) are highly useful to measure the integrity of an entire computer system in a trustworthy manner. Unfortunately the integrity verification performed by an external verifier does not fit into our scenario nicely, which we want to explain in the following.

An additional computer system that acts as verifier could be deployed in the local network and communicate the result of the integrity verification to the human user via email, for instance. However, the usability of this approach seems to be impaired and additionally another machine would be needed. Moreover, in order to trust the assertion given by the verifier, the verifier's integrity itself must be proven to the user. This creates a chicken-and-egg problem.

Besides the described drawback for our scenario, Remote Attestation concepts generally have privacy problems. The reason for this is that the entire configuration of the verified computer system is sent to the verifier. An attacker that has compromised a verifier might deduce from loaded applications special interests of the system owner or, in the worst case, abuse the knowledge gained about the system's configuration to search for weaknesses, e.g., applications with known vulnerabilities.

12.3.3 Synthesis

Following Table 12.2 concludes the above-described considerations. Neither integrity verification by classic Remote Attestation nor both described Secure Boot concepts are suitable for our scenario. Furthermore, both approaches have specific drawbacks, such as privacy concerns (Remote Attestation) or the quite incomplete integrity verification of system elements (Secure Boot). Nevertheless, the integrity measurements of an entire computer system based on the extended Trusted Boot appear to be beneficial.

	Secure Boot	**Remote Attestation to Server**
Trustworthiness	✗ User does not know if SB is used and integrity of device checked.	✗ External server needed. User does not know if this server has integrity itself.
Verification Completeness	✗ Only first components in boot chain are verified.	✓ Integrity measurement can be extended up to user space.
No Verification Server Needed	✓ Software on host verifies integrity.	✗ Software on external server verifies integrity. Problematic in our scenario.
Preserves Privacy	✓ No information about device communicated to the outside.	✗ Detailed information about device communicated to the outside.

Table 12.1: Suitability of the state of the art of integrity verification mechanisms to our scenario.

12.4 Design

In this section the approach of our solution is first outlined on a high level. Later, the introduced mechanisms are described in detail.

12.4.1 Approach

The overall solution to guarantee the trustworthiness and integrity of Domain Server and virtual machines to users consists of four, below-explained building blocks.

Integrity Verification of Virtual Appliances

The Domain Server Manager, which is the service that manages and starts Security Service or Service VMs on behalf of a Domain Owner, is extended by a function that verifies the authenticity and integrity of virtual appliances. For this purpose the maintainers of virtual appliance must certify their products. After verifying the validity of a virtual appliance,

the Domain Server Manager is confident that a just downloaded or already stored virtual appliance is authentic and trustworthy and starts a VM using the verified image. If the verification fails, the image will not be used.

The PKI infrastructure required by the proposed mechanisms is detailed in Section 12.4.2. The workflows needed to verify a virtual appliance are described in Section 12.4.3.

Protection of Persistent Virtual Machine Images

The Domain Server Manager is extended with a new functionality that allows creating encrypted containers for VMs that need to be stored persistently on the Domain Server. The public part of an asymmetric key pair contained in the smart card of a Domain Owner encrypts ("wraps") the symmetric key (volume key), which in turn is used to encrypt the volume. For this reason, a persistent VM can only be loaded if the smart card that contains the private key needed to decrypt the volume key is present and allows the decryption.

The workflow that creates persistent VM images from verified virtual appliances is detailed in section 12.4.3.

Integrity Verification of Domain Server

As we have just discussed, the state of the art does not offer any solution suitable to prove the integrity of the Domain Server to a Domain Owner. We propose a novel approach based on concepts taken from the state of the art to prove the integrity of the Domain Server to an Integrity Verifier applet contained and executed inside a Java smart card. This card is trustworthy, as it is owned and controlled by a Domain Owner, see Figure 12.1, Step 1. For an introduction of Java smart card technology see Section 9.2.2. The initial trust established into the Domain Server (Steps 2 and 3) is then used to create a chain of trust to above described functions for integrity verification of virtual appliances and to the functions that handle encrypted VM volumes (Steps 4 and 5).

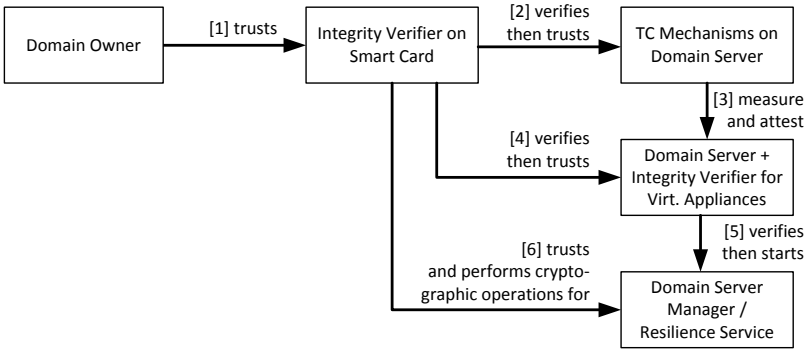


Figure 12.1: Establishing trust into the Domain Server and started virtual appliances.

For measuring the integrity of the Domain Server, the extended TCG Trusted Boot (CRTM, Trusted Boot Loader and IMA kernel extension) will be used. The fingerprints of loaded system components are communicated from the Domain Server to the Integrity Verifier applet using the TPM-supported Remote Attestation protocol. The communication is performed over the TLS/SSL protected HTTP-connection provided by the used Java smart card. These mechanisms are detailed in section 12.4.5.

Decryption of Private Data after Domain Server Integrity Verification

After verifying the integrity of the Domain Server, the Integrity Verifier is confident about the trustworthiness of the Domain Server and its functions, i.e., the correct verification

of virtual appliances and handling of encrypted volumes. For this reason, the Java smart card will agree to perform cryptographic operations needed to decrypt private data of the Domain Owner on behalf of the Domain Server Manager or the Resilience Service running within a Security Service VM (Figure 12.1, Step 6). The interaction of all outlined mechanisms is detailed in section 12.4.4.

12.4.2 Certification of Involved Entities

In following chapters mechanisms are introduced that require various certificates types to verify the validity of asymmetric keys, signatures of virtual appliance, involved Java smart cards and TPMs. These certificates, their purposes and their issuers are introduced and explained in the following.

A *Maintainer CA* issues a *Virtual Appliance Certificate*. In contrast to other certificate types introduced later, no key but the cryptographic hash value of the virtual appliance is certified. The Virtual Appliance Certificate is used by the Domain Server Manager to validate the authenticity and integrity of a virtual appliance before the appliance is used.

A *Card Manufacturer CA* issues a *Card Certificate*, which expresses that the certified key is contained inside a valid Java smart card equipped with an Integrity Verifier applet. The Card Certificate is used by the Attestation Service to validate if a Java smart card is authorized to perform the integrity checks.

A *Privacy CA* issues an *AIK Credential*. The credential expresses that a specific AIK private key was generated in and is inseparably bound to a specific TCG compliant TPM chip. This certificate is used by the *Integrity Verifier* on the Java smart card to establish a basic level of trust into the platform that is about to be integrity checked.

The described certificates can either be signed by the same CA, by completely independent CAs, or by CAs that belong to a CA hierarchy. This PKI can be rooted in the CA of the Domain Server manufacturer, see Figure 12.2.

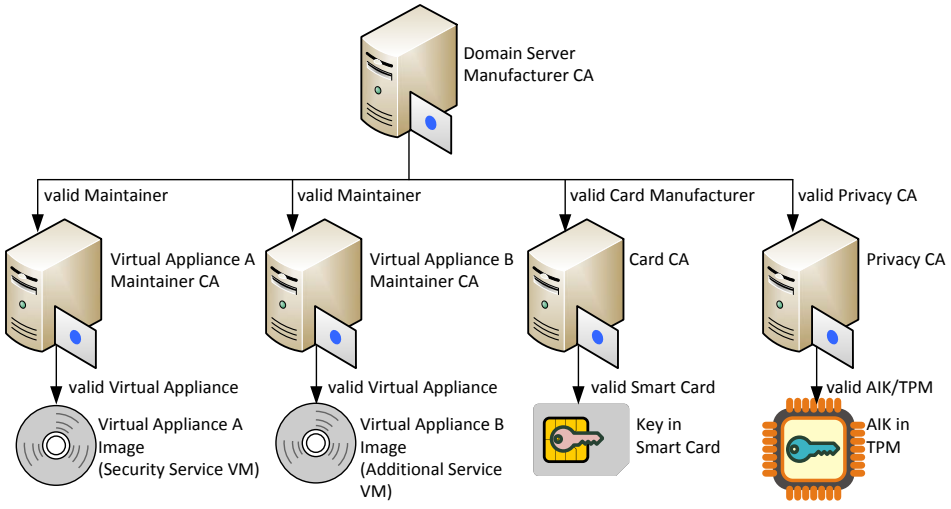


Figure 12.2: CA hierarchy for validation of signatures and keys.

In our opinion, a hierarchic approach is most beneficial as it simplifies certificate distribution, as only the root certificate of the Domain Server manufacturer must be supplied to Domain Servers and Java smart cards to allow them to verify certificates created by CAs that belong to this PKI.

The certificate issued by the Domain Server manufacturer's Root CA for a Card CA is called *Card Manufacturer Certificate*; the certificate issued for a virtual appliance Main-

tainer CA is called *Maintainer Certificate*; finally the certificate issued for a Privacy CA is called *Privacy CA Certificate*.

12.4.3 Manufacturing and Import of Virtual Appliances

Manufacturing of Trustworthy Virtual Appliances

The maintainer of a virtual appliance must first obtain a Maintainer Certificate from the Domain Server manufacturer, see Step 1 of Figure 12.3. The assessment procedure required to prove the trustworthiness of maintainer to server manufacturer is out of scope of this thesis. If the manufacturer was able to prove her trustworthiness to the Domain Server manufacturer she obtains a Maintainer Certificate for her own CA (Step 2), which later issues Virtual Appliance Certificates.

Figure 12.3: Manufacturing workflow of trustworthy virtual appliances.

The maintainer can now create her virtual appliance (Step 3) and issue a Virtual Appliance Certificate that contains the virtual appliance's fingerprint (Step 4). The Virtual Appliance Certificate expresses that the maintainer has checked all programs contained in the virtual appliance, i.e., that the virtual appliance is trustworthy. Finally the virtual appliance, the Virtual Appliance Certificate and the Maintainer Certificate are made available for download (Step 5).

Please note: The described process does not differ between virtual appliances that contain the Guided Security Management System or virtual appliances that contain other services.

Import and Usage of Trustworthy Service VMs

In this paragraph the import of a virtual appliance of a Service VM into an encrypted volume is described. Virtual appliances of Security VMs are handled differently, see Section 12.4.4.

First, the virtual appliance and the certificates that prove the authenticity and trustworthiness of this image are downloaded to the Domain Server, see Figure 12.4, Step 1. The Domain Server Manager now verifies the virtual appliance using the supplied certificates (Step 2). If the Domain Server Manager is convinced of the trustworthiness of this image, it prepares the encrypted volume that later stores the just verified service persistently. Otherwise, if the verification fails, the image will not be used.

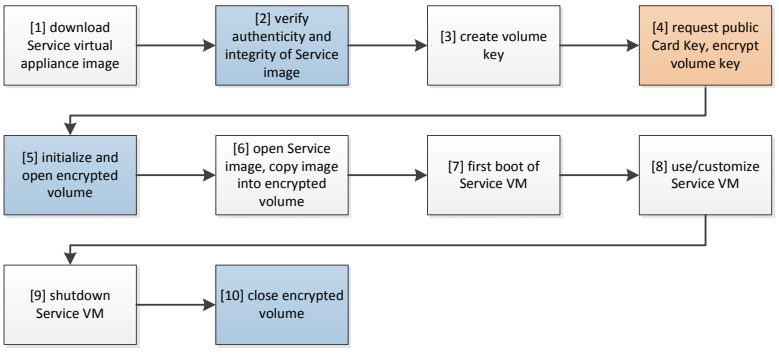


Figure 12.4: Validation and import of a trustworthy virtual appliance into an encrypted persistent volume.

Actions performed by the Domain Server to verify the integrity of virtual appliances or to create an encrypted, persistent VM image are colored in a blue tinge (Steps 2, 5, 10). Actions that involve the Domain Owner's Java smart card are colored in an orange tinge (Step 4).

For storing the service in a persistent, encrypted volume, a new symmetric key, the volume key, is generated (Step 3). The Domain Server Manager requests the public part of the Card Key, which is only exported from the Java smart card if the previous integrity verification of the Domain Server could be completed successfully. This public key is then used to encrypt the volume key (Step 4).

Now the Domain Server Manager creates a new encrypted volume using the volume key and opens it (Step 4). The content of the verified virtual appliance is then copied into the volume. With this step, the preparation of the securely and persistently stored Service VM is finished.

The VM is booted for the first time now (Step 7) and can be customized by its owner (Step 8). At a later point in time, when the Domain Owner unplugs her Java smart card from the Domain Server, the running Service VM is shut down (Step 9) and the volume is closed (Step 10).

12.4.4 Decryption of Private Data After Integrity Verification

The workflow depicted in Figure 12.5 describes how all mechanisms interact in order to start a Security Service VM and to start a persistent Service VM stored in an encrypted volume. This workflow is based on the workflows for Automated Domain Management and Domain Resilience, which were already described in Section 11.3.4.1. The actions, which are not directly relevant in this context, are omitted for brevity.

Again, actions performed by the Domain Server to verify the integrity of virtual appliances or to open an encrypted, persistent VM image are colored in a blue tinge (Steps 6, 8). Actions that involve the Domain Owner's Java smart card are colored in an orange tinge (Steps 3, 5, 12).

Domain Server Boot Process and Integrity Verification

The Domain Server is booted using the extended Trusted Boot mechanisms described above (Step 1). As in the original workflow, the Domain Server Manager is automatically started after the Domain Server has finished its boot process and waits for Java smart cards to be plugged in (Step 2). When a Java smart card is plugged into the Domain Server, the Domain Server Manager starts the integrity verification process (Step 3), which is detailed in Section 12.4.5.

The Domain Server Manager now starts all VMs needed by this Domain. First the Security Service VM of the Domain is started. Next, persistent VMs are started.

Start and Validation of a Security Service VM

To start a Security Service VM, the Domain Server Manager first verifies the authenticity and integrity of the Security Service virtual appliance (Step 8), and then starts the Security Service VM (Step 9).

After the boot process of the Security Service VM has finished, the Resilience Service is started automatically (Step 10) and performs its tasks. The Resilience Service downloads the encrypted state backup that belongs to this domain and the encrypted backup key, which is needed to decrypt the backup (Step 11). Now the decryption of the backup key

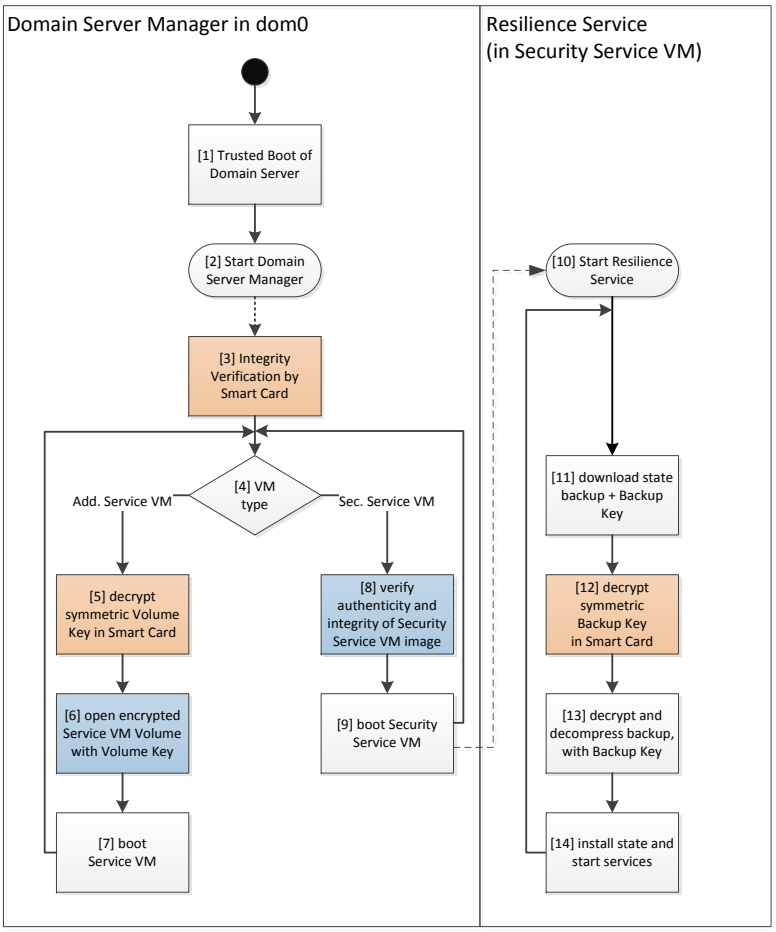


Figure 12.5: Interaction of Domain Server Manager and Resilience Service enhanced with functionality for integrity verification and protection of persistent VM images.

using the private part of the Card Key contained on the Domain Owner's Java smart card is requested. Only if the Integrity Verifier applet is confident about the trustworthiness of the Domain Server and its integrity verification functionalities, the Java smart card will decrypt the backup key (Step 12). The Resilience Service now decrypts and installs the Domain's state (Step 13) and starts all services that belong to the Security Service VM.

Start of a Persistent Service VM

After starting the Security Service VM of a Domain, previously generated persistent Service VMs are started. For this purpose the Domain Server Manager requests the Java smart card to decrypt the volume key that encrypts the VM image (Step 5). Again, the Java smart card will only accept this request, if the Domain Server's integrity could be verified before successfully. The volume key is then used to open the encrypted VM Volume (Step 6). Finally, the Service VM is started (Step 7).

12.4.5 Integrity Verification of a Domain Server

12.4.5.1 Preparatory Work

For the attestation protocol to work, some preparations are needed to customize both the Java smart cards and the Domain Servers.

Java Smart Cards

The manufacturer of the Java smart cards that contains the Integrity Verifier applet must undergo a similar procedure as described for maintainers of virtual appliances in order to obtain a Card Manufacturer Certificate for the own CA. The card manufacturer must then deploy the Integrity Verifier applet to the Java smart card, create a *Card Key* on this card and finally certify this card's validity (more precisely: the Card Key) using her CA. Both, the Card Certificate and the Card Manufacturer Certificate need to be stored on the Java smart card.

Domain Server

For performing the Remote Attestation, the Domain Server must be equipped with an Attested Identity Key (AIK) and an AIK credential. This credential is issued by a Privacy CA that belongs to the used PKI. Both, the Privacy CA Certificate and the AIK credential need to be stored on the Domain Server.

Binding Java Smart Cards to a Specific Domain Server

It is possible to "bind" one or several Java smart cards to a specific Domain Server. For this purpose a Java smart card must be additionally equipped with the public part of the AIK of the Domain Server it should be tied to.

If this option is chosen, a Java smart card will accept only a specific Domain Server. The benefit of this binding is that it is impossible to replace a Domain Server with another, maybe manipulated, machine. The drawback is that only one specific Domain Server can be used. So, for instance, when a Domain needs to be moved to another home, this is impossible as the Java smart card will refuse to decrypt private data of the Domain on the "unknown" Domain Server.

A better option is to only use the Privacy CA Certificate to verify if the Domain Server is equipped with a TPM that stores a valid AIK. The Integrity Verifier applet on the Java smart card will then be able to perform the remote attestation protocol with any Domain Server.

12.4.5.2 Attestation and Verification Protocol

Figure 12.6 depicts the Remote Attestation of a Domain Server and its verification by the Integrity Verifier applet running inside a Java smart card. At the point in time this protocol is performed, the Domain Server has already finished its Trusted Boot process, i.e., fingerprints of all executed system components are created and logged to the IML, which is protected by the IVV contained in the Domain Server's TPM.

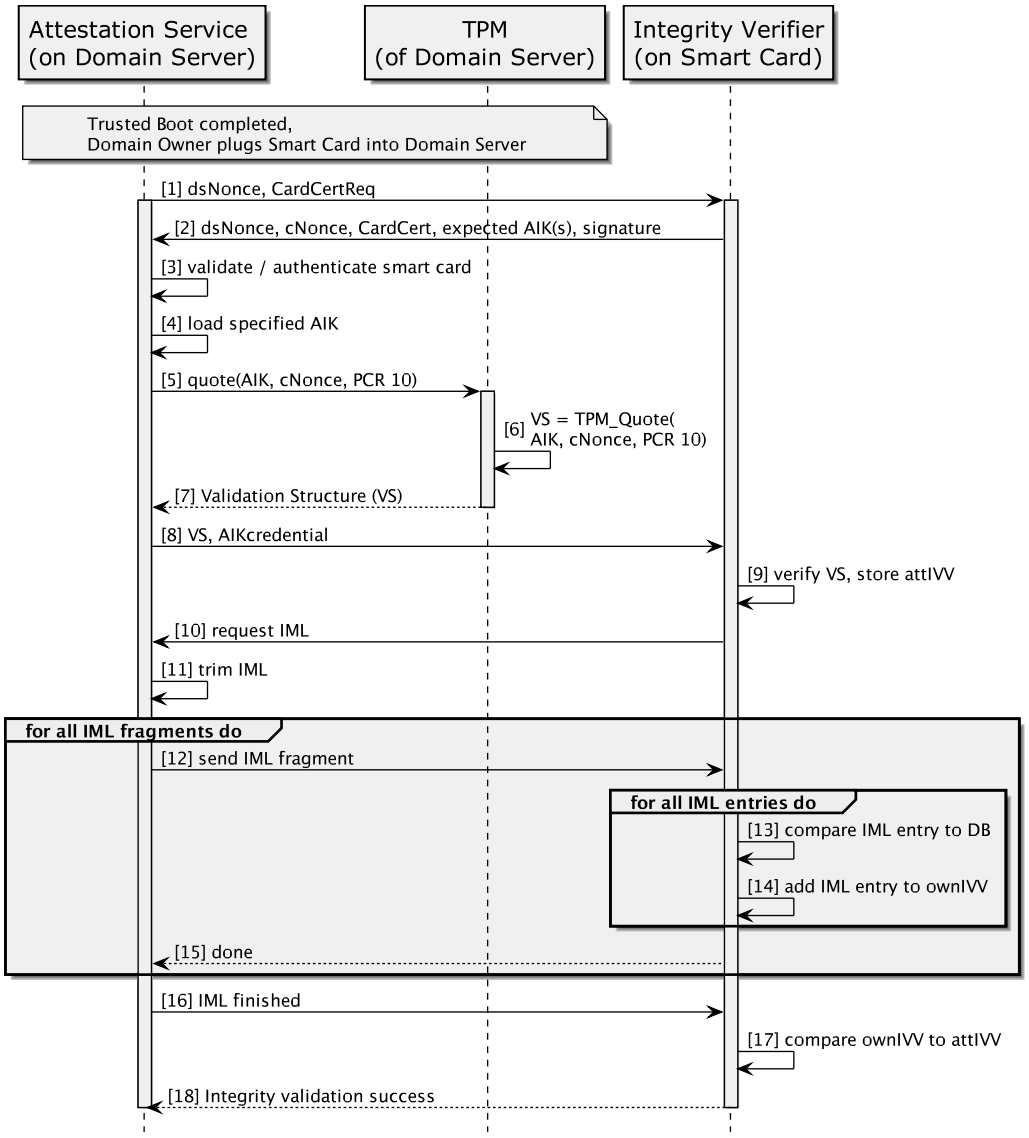


Figure 12.6: Remote Attestation of Domain Server to Java smart card and Domain Server integrity validation.

After plugging a Java smart card into the Domain Server, the Domain Server Manager activates the *Attestation Service*. The Attestation Service creates and sends a random nonce (dsNonce) and a Card Certificate request (CardCertReq). The nonce is used as a challenge for a simple authentication protocol that proves the validity of the Java smart card's verifier applet to the Attestation Service.

The Integrity Verifier applet contained on the Java smart card generates a random value (cNonce) for replay protection of attestation data. Optionally, an AIK that is expected to later sign the Validation Structure can be specified. This is done by either directly specifying the AIK to use or indirectly by specifying a trusted Privacy CA. Finally the In-

tegrity Verifier generates and signs the message that contains Card Certificate (CardCert), cNonce and dsNonce. The message is signed using the Card Key (Step 2).

In Step 3, the Attestation Service verifies the validity of the Java smart card by validating if CardCert was signed by a trustworthy Card Manufacturer CA, if the signature of the received message was signed by the private part of the corresponding Card Key, and if dsNonce was included in this signature.

Before we continue with the explanation of the protocol we want to point out a limitation of the Java smart cards concerning Remote Attestation. The standard Remote Attestation protocol sends IML and IVV to the verifier in one operation. This is not possible in this case, as Java smart cards have a limited amount of main and persistent memory only. Hence, they cannot process or store the entire set of data at one time. For this reason, IVV and IML need to be sent to the Java smart card separately. Depending on the Java smart card's main memory it might even be necessary to split the IML into fragments, which are then transported step by step to the Integrity Verifier and processed by it. This results in a process that will take some time and the possibility that applications might be loaded and measured concurrently is increased. This will lead to an IML that does not correspond to the already sent IVV anymore. Finally, the integrity verification of the Domain Server will fail for this reason.

The TPM `quote` operation for the IVV, which is contained in PCR 10, is prepared by loading a suitable AIK into the TPM (Step 4). Now the IVV contained in PCR 10 is signed using this AIK. For replay protection, cNonce previously provided by the Integrity Verifier is included into the signed Validation Structure (VS) (Steps 5 - 7). Finally, VS is sent to the Integrity Verifier applet on the Java smart card together with the credentials of the used AIK (Step 8).

The Integrity Verifier checks the validity of VS in Step 9, i.e., checks if the used AIK's credential is valid and appropriate. Finally, the Verifier checks if the AIK signature of VS is valid and if cNonce was included in VS. If the verification is successful, the Integrity Verifier is convinced that the attested IVV is valid and saves IVV for later reference. Finally, the verifier requests the IML from the Attestation Service in Step 10.

The Attestation Service trims IML in Step 11. First all parts of the IML, which are not absolutely necessary (e.g., the paths to the loaded applications) are removed from the IML to spare memory on the Java smart card and to simplify the processing. The result of this operation is a pure list of fingerprints of loaded applications. This step does not reduce the credibility or meaning of IML, as only the fingerprints of executed software are needed to verify the Domain Server's integrity. The Attestation Service now computes the IVV over the fingerprints using the recursive `extend` algorithm explained in Section9.3.2.4. The result of this operation is compared to the IVV already sent to the Java smart card. If there is a mismatch, an application was executed and measured in the meantime most probably. Therefore, the Attestation Service tries to rectify the IML by removing successively the latest entries from IML until IML matches IVV. If this operation fails the IML has been tampered and the process fails.

The trimmed IML is sent in smaller fragments to the Integrity Verifier applet. Steps 12 - 15 need to be repeated until all fragments of the IML are transferred to the Java smart card. In Step 13 the Integrity Verifier applet compares a fingerprint taken from the IML fragment to the local database of trustworthy applications (white list). If this fingerprint could not be found, the integrity verification fails as the verifier assumes that either an unauthorized or a tampered application was loaded on the Domain Server. Additionally, the Integrity Verifier applet computes her own IVV (ownIVV, Step 14) using the recursive `extend` algorithm, which is later compared to the IVV received in Step 9. When all fingerprints are processed, the verifier applet requests the next IML fragment (Step 15).

If all fragments of IML are transferred to the Integrity Verifier applet, a corresponding message is sent in Step 16. The verifier now compares the locally computed ownIVV to the received attIVV (Step 17). If both values match, the verifier is confident that the received IML was trustworthy. With this step the integrity verification of the Domain Server is successfully finished.

A success message is sent in Step 18 to the Attestation Service, which terminates the process.

12.5 Further Application Example: Secure VPN Access

The just presented technology for integrity verification of a device using a trustworthy software component contained on a Java smart card is highly useful in other contexts as well. As an example we have integrated the Java smart card-based integrity verification into the IKE protocol performed between VPN client and server. Besides authenticating the user that requests network access using the private key contained on her Java smart card, the access server is able to additionally gain confidence about the trustworthiness of the accessing device. This is highly useful to prevent that devices that execute unauthorized software access the network.

12.5.1 Platform Verification Certificate

To prove the successful integrity verification of a device by an Integrity Verifier applet contained in a Java smart card to some other entity, we created the *Platform Verification Certificate*. The PVC is assembled and signed by the Integrity Verifier applet after finishing the integrity validation of the device in Step 17 of Figure 12.6. The PVC connects the identity of the verified device (represented by this platform's AIK) and the Identity of the verifying Java smart card (represented by the Card Key).

In Figure 12.7 the contents of the PVC are depicted. The single fields of this data structure contain the following information:

- **Attestation Type and Version**: Fields used to specify how the system was verified. Currently only one verification type is implemented (verification using integrity measurement values). Other types or versions using different algorithms or different data sets might be supported in future as well.
- **Validity:** Timestamps that express in which time frame the PVC is valid.
- **Database Version:** The version of the database (white list) containing fingerprints of authorized applications stored on the Java smart card. This information is important as databases might be replaced by updated versions, which render the old versions invalid.
- **Card Certificate:** The Card Certificate (identity) of the Java smart card that verified the device and signed the PVC.
- **AIK Credential**: The AIK Credential (identity) of the device that was integrity checked.
- **Signature:** The signature created by the Java smart card over the previous fields using the private part of the Card Key.

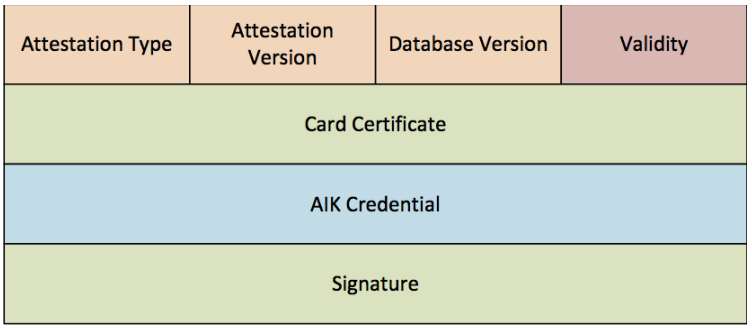


Figure 12.7: Platform Verification Certificate

12.5.2 Extending Authentication Protocols with Integrity-Awareness

In Figure 12.8 the integration of the PVC into the authentication protocol between VPN client and server is shown. The depicted protocol is a slightly modified IKE handshake as described RFC 5996 [88, Section 1.2].

The VPN client installed on the device requests access from the VPN server (Step 1). If specified by the local Security Policy, the VPN server sends a PVCREQ, a request for a Platform Verification Certificate to the client (Step 2). A further important, yet standard, part of this message is a nonce (Nr), which can be regarded as challenge in this authentication protocol.

Besides static PCRs, such as PCR 10, which contains the IVV used as checksum of the IML of a device, TPMs offer various dynamic PCRs, which can be reset on demand and extended with arbitrary data. Such a dynamic PCR is used to bind the response to the challenge previously received from the VPN server to the device identified by its TPM that requests network access. For this purpose, the pieces of information received previously are first hashed (Step 3). The resulting hash value is then extended into the dynamic PCR 17 (Step 4). Subsequently, the hash value now contained in PCR 17 and Nr are `quote`d (signed) using the AIK referenced in the PVC.

The result of this `quote` operation, the Validation Structure VS, is now signed by the Java smart card using the private part of its Card Key (Steps 9 - 11). The purpose of this step is to create a data structure (SigVS) that binds the VPN server's challenge to the Java smart card.

The VPN client now generates PVCAUTH, which consists of VS from Step 7 and SigVS from Step 10. As final answer to the server's challenge sent in Step 2, the PVC generated by the Smart Card and PVCAUTH are sent (Step 13).

The AP verifies the received information as follows (Step 14): First, the validity of the Java smart card involved in this process is validated by verifying the Card Certificate contained in the PVC using the Card Manufacturer Certificate. Now the AIK Credential contained in the PVC is verified using the Privacy CA Certificate. Finally the public Card Key contained in the Card Certificate is used to verify the signature of the PVC. After finishing these steps, the VPN server is confident to have obtained a PVC issued by a valid Java smart card for a valid TCP that only runs authorized software.

Now the VPN server extracts VS from PVCAUTH. It first verifies the freshness of VS by checking if the expected nonce Nr is contained in the signature and if the AIK that has signed VS is the same as referenced in the PVC. Finally the verifier checks sigVS, which should be signed by the Card Key. These steps connect the access request to the Java smart card that has signed the PVC, which expresses the platform's integrity. These steps

also guarantee that the verified platform is the same device that now tries to access the network.

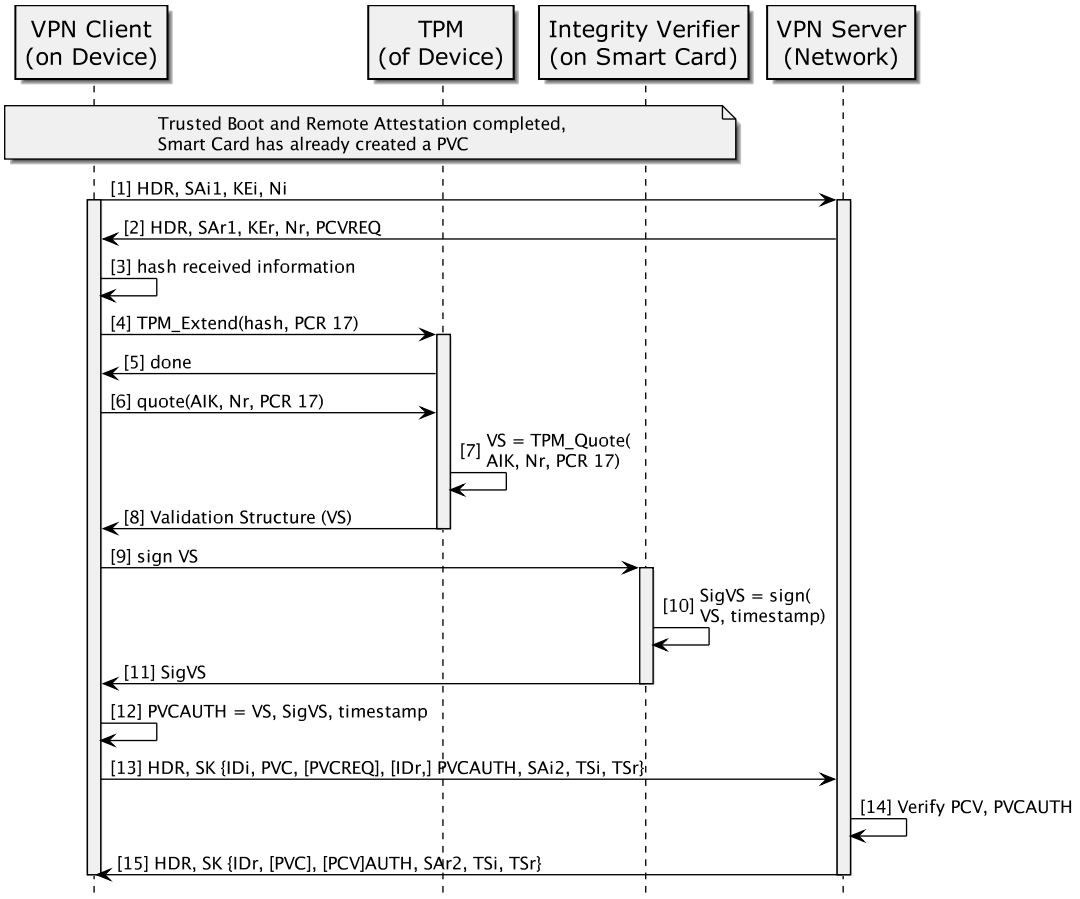


Figure 12.8: Integrity-aware Internet Key Exchange protocol.

12.6 Discussion and Evaluation

12.6.1 Fulfillment of Requirements

The mechanisms that ensure trust and integrity of Domain Server and virtual machines fulfill all requirements we have defined in Section 12.2.

The required proof of the Domain Server's trustworthiness to the Domain Owner (RE.1) works implicitly. First, the Domain Server's integrity is proven to the Integrity Verifier applet on the Java smart card. If the Java smart card is convinced about the Domain Server's integrity, it will allow cryptographic operations on behalf of software components running on the Domain Server. These operations are needed to decrypt the encrypted VM volumes that contain personal data of the Domain or to decrypt the Domain's state (RE.4/RE.5). If the VMs of a Domain are successfully started, the Domain Owner can be confident that the Domain Server is trustworthy.

The proposed concept is comparable to the conventional Secure Boot mechanism specified by TCG, as both systems allow the decryption of data only if the already loaded system components have integrity. But our solution is superior to standard TCG Secure Boot, as the users of our system can trust the proposed mechanism. The reason for this is that we use the Java smart card, which is owned and controlled by the user, to verify the system. Therefore, the user can be confident that the card will perform the integrity verification of the Domain Server as intended. In contrast to the user, the Integrity Verifier applet

on the Java smart card does not need to initially trust the Domain Server. The reason for this is that the verifier is able to establish trust into the Trusted Boot functions of the Trusted Computing Platform by evaluating the platform credentials. Then the verifier gradually validates the transitive chain of integrity measurements rooted in the CRTM of the TCP, which finally proves the trustworthiness of the verified system. The user of a system controlled by somebody else that uses conventional Secure Boot does not know for sure if Secure Boot is used as intended on the system. Therefore, the user cannot trust this system.

Based on a simple but efficient signature scheme for virtual appliances rooted in the CA of the Domain Server manufacturer, trust into the integrity and authenticity of virtual appliances (RE.2) can be established. For this purpose the Domain Server Manager verifies the authenticity of images using Virtual Appliance certificates and Manufacturer Certificates before their usage (RE.3).

Finally, the proposed Java smart card-based mechanism is user-friendly and does not impose any drawback to the users (RE.6). The reason for this is that the concepts proposed in this chapter could be integrated seamlessly into the already described mechanisms. Furthermore, the Domain Owners already own a personal cryptographic token, which was so far only used to contain a cryptographic key and to perform cryptographic operations on request by the Resilience Service running inside a Security Service VM. Now the simple token is exchanged with a sophisticated Java smart card that performs integrity checks. The usability of the system and the way the user interacts with the system remain the same.

The only difference for a Domain Owner that results from the newly introduced integrity verification mechanisms are slower response times. So when the user plugs in his Java smart card a longer timespan will pass until the VMs of a Domain are started. However, this drawback can be neglected as the user will not shutdown and restart her VMs too often.

12.6.2 Further Benefits

The integrity verification of a Trusted Computing Platform by the Integrity Verifier applet can be easily integrated into any authentication protocol such as TLS, OpenVPN, etc. As we have shown, the integration is quite simple to achieve, as only a small amount of messages of the original protocol flow need to be extended. Moreover, standard functions that authenticate the accessing entity need to be extended with the ability to cope with the newly added data structures and the PVC that prove the successful integrity verification.

The use of Java smart cards for integrity verification is highly beneficial as there is no need for a central integrity verification server anymore. The quite complicated verification of device integrity is distributed and performed by many Java smart cards. The data structures and PVC generated by the Java smart card, which prove the successful integrity verification, are quite simple to verify. For this reason, no bottleneck can occur with the Java smart card-based concept.

The proposed integrity verification by Java smart cards is also free of privacy concerns, as no data needs to be sent to a central verification server that would reveal the devices configuration.

The properties of the proposed technology are subsumed in following Table 12.2 and compared to the state of the art.

	Secure Boot	Remote Attestation to Server	**Remote Attestation to Java Smart Card (This Thesis)**
Trustworthiness	✗	✗	✓ User trusts Java smart card, which will only decrypt information when the Domain Server's integrity could be proven.
Verification Completeness	✗	✓	✓ Complete verification of measurement values up to user space possible.
No Verification Server Needed	✓	✗	✓ The trusted Java smart card performs integrity verification.
Preserves Privacy	✓	✗	✓ Preserves privacy of the device as no information about its configuration is revealed to the outside.

Table 12.2: Comparison of the State of the Art to the Proposed Technology. (Explanation of columns "Secure Boot" and "Remote Attestation" omitted for brevity, see Table 12.2.)

12.6.3 Limitations of Java Smart Card Hardware

Despite the fact that the specification of JavaCard Version 3.0 "Connected Edition" is already several years old, no card hardware that features this technology could be purchased at the point in time (late 2012) when this research work was performed. For this reason, the described verification mechanisms were developed and tested on a Java smart card simulator that implements the new JavaCard standard.

As this simulator is only available for Microsoft Windows operation systems, an additional simple communication proxy had to be used to establish the communication between the verified Linux system and the simulator.

At a later point in time an early prototype of the required card technology was provided to us from a well-known manufacturer of smart card hardware. This prototype was not usable due to severe driver problems.

12.6.4 Limitations of Trusted Computing Technology

One important limitation of Trusted Computing technology is the lack of Platform Credentials. Both Lenovo laptops manufactured in 2009 we used for this work were not equipped with the required credentials. This is especially surprising as Lenovo is member of the TCG. The lacking credentials become problematic in case AIK Credentials need to be obtained from a Privacy CA.

The availability of Privacy CAs is problematic, too. To the knowledge of the Author, the only publicly available Privacy CA at this point in time (mid 2013) is *PrivacyCA*. The operator of this service clearly states that *"Privacy CA is at this time an experimental technology demonstration and is exclusively for personal and experimental use."* [89] Privacy CA will issue AIK Credentials also to those device not equipped with Platform Credentials.

However, these examples are no limitation of the Trusted Computing technology but shortcomings at the sides of system manufacturers.

12.6.5 Applicability

The mechanisms we described are generic and can be applied in different contexts as well. The integrity verification by Java smart cards can be used, for instance, for kiosk computing. When a user of this system plugs her Java smart card into a host she can trust this system and be sure that her private data is only revealed to the system when it has integrity.

Furthermore, the developed Integrity Verification approach can be integrated into legacy authentication protocols in order to make these protocols aware of the integrity of the accessing device quite easily. In contrast to the state of the art, our solution will protect the privacy of the accessing device.

12.7 Conclusion

The Domain Server is a system that uses virtualization to isolate different stakeholders and different services. For this reason the trustworthiness and integrity of the Domain Server (hypervisor and Xen dom0) and of virtual machines (Xen domUs) the Domain Server hosts can be considered as two separate problems.

We discussed that the creation and deployment of faked virtual appliances is a rewarding attack vector on our system. Modified versions of our software might be distributed and might lever out the security and effectiveness of our access control system. For this reason

we proposed a simple and effective mechanism how images of virtual appliances can be signed by their maintainers and their integrity and authenticity verified by the Domain Server Manager.

Besides the distribution channel, persistently stored VM images can be attacked in a different way, namely when they are stored without further protection on the Domain Server or sent over the network to another Domain Server. For this reason we have integrated volume encryption for persistent images that uses keying material controlled by software housed in a Java smart card. This software only allows the decryption of data if it could verify the integrity of the Domain Server before.

For guaranteeing the trustworthiness and integrity of the Domain Server we first evaluated the state of the art but found no technology that is completely satisfying in our scenario. Hence, we combined suitable Trusted Computing technologies that measure and report the integrity of a system in a trustworthy manner to an Integrity Verifier applet housed inside a trustworthy Java smart card. If the Integrity Verifier in the card is convinced of the trustworthiness of the verified system, which also includes the establishment of trust into just described functions that authenticate or protect virtual machines, the Java smart card will perform cryptographic operations needed to decrypt valuable personal information of Domain Owners on the Domain Server.

Key Findings and Contributions

▷	Integrity violations of the Domain Server itself or of images of virtual appliances that contain the Guided Security Management System for Domains or other services are of major concern.
▷	Therefore, mechanisms are needed that guarantee the integrity of the Domain Server and virtual appliances.
C12.1	The **Integrity Verifier applet** located on the Domain Owner's Java smart card evaluates the integrity of a Domain Server based on integrity measurement values created by Trusted Computing technology.
C12.2	The **Domain Server Manager extension** evaluates the integrity of virtual appliances based on a signature scheme before their start.
C12.3	The combination of Domain Server Manager/Resilience Service and Integrity Verifier applet enforces that personal data of a Domain Owner can only be decrypted when the the execution environment has integrity.
C12.4	The **Platform Verification Certificate (PVC)** makes existing authentication protocol aware of the integrity of an accessing entity.

13. Security and Trust for Private Keys of Domain CAs and Entities

A basic requirement of every reliable and secure access control system based on asymmetric cryptography is the secrecy of private keys used by all involved parties. As the access control system proposed in this thesis uses asymmetric cryptography, the secrecy of private keys becomes highly important to us.

In the previous chapters various mechanisms targeted to the overall security and trustworthiness of the Domain Server were introduced. These mechanisms already have some positive effects on the secrecy of private keys used by the Domain CAs to sign certificates. For instance, one important goal of the Domain Server architecture was the thorough isolation of different stakeholders. Inter alia, this will avoid that one stakeholder is able to easily access another stakeholder's Domain CA, which would allow her to abuse this CA. Also the Resilience Service was designed with the secrecy of private keys of Domain CAs in mind. This goal could be achieved by encrypting the Domain state backup, which contains the private key of a Domain CA, using an asymmetric key contained inside the smart card that belongs to the Domain Owner.

However, these mechanisms only target some aspects of key security and are limited to the Domain Server only. For this reason, we do not regard the overall requirement RF "Protection of Keying Material" as being fulfilled yet. Hence, this issue will be addressed in the following.

Chapter Structure

We first analyze in Section 13.1 various threats that arise when private keys, either of Domain CAs or entities that belong to a Domain, are stolen by an adversary or abused by their legitimate owners. Consecutively, we define in Section 13.2 requirements on a mechanism able to protect the private keys in order to increase the security of our access control system. In Section 13.3 we discuss the applicability of various existing hardware-based solutions for key protection to our scenario. The integration and adaptation of the most promising base technology into the existing system is then detailed in Section 13.4. Finally, we evaluate the properties of the hardened access control system in Section 13.5 and conclude the chapter in Section 13.6.

Acknowledgments

This chapter contains results developed in the Bachelor's Thesis of Simon Mittelberger [90] and results from Simon Stauber's and Simon Mittelberger's work as student research assistants. These works were assigned and supervised by the Author in the context of his doctorate.

13.1 Threat Analysis

In the following we examine which negative effects emerge from insufficiently protected private keys on the reliability and security of our access control system. The analysis is split into two parts: negative effects that occur after identity theft and negative effects caused by key abuse by their legitimate owners.

13.1.1 Identity Theft

The most obvious threat for our access control system is when the private key of a Domain CA or of an entity that belongs to a Domain gets stolen. As such a private key represents the identity of a complete network (Domain CA assigned to a home or company network), a user (Domain CA assigned to a person) or an entity assigned to a Domain, we refer to this attack type as *identity theft*. Consequently, we refer to the asymmetric key pair of a Domain CA or an entity as *Identity Key (IK)*.

The following analysis is performed from the perspective of the legitimate owner of the stolen IK and the perspective of other Domains that share a Trust Relationship with the compromised Domain.

Theft of a Domain CA's Identity Key

For an attacker or malware that gained access to the environment a Domain CA resides, i.e., the Security Service VM, it is quite simple to extract the Domain CA's IK and send it to the adversary over the network. Although it is possible to protect a key with a PIN or password, the attacker can easily break this protection without too much effort. She might either brute force the PIN/password[1] or she might have figured out the key's PIN/password using a key logger. Therefore, standard protection methods of keying material must be regarded as insufficient.

The result of identity theft of a Domain CA is that the attacker is able to issue "bogus" certificates that cannot be distinguished from authentic certificates issued by the legitimate Domain CA. Once the attacker has equipped her own entity with a bogus certificate, she is able to either abuse services offered in the Domain she has stolen the IK from or she is able to abuse services shared by other Domains with the compromised Domain. This ability is especially displeasing, as many other Domains are affected, too.

As it is quite difficult to detect identity theft, it is possible that the subsequent abuse of services, either in the compromised Domain or in other Domains, remains undetected for quite a long time. This is problematic.

Once the identity theft is detected, the situation is still quite simple to resolve when no Trust Relationships to other Domains exist. Only a new IK must be generated for the Domain CA and all entities that belong to this Domain must be newly registered in order to obtain a certificate signed by the new Domain CA's IK.

[1] Brute forcing the PIN/password of a private key seems to be feasible considering the performance of modern CPU/GPU hardware. Additionally, there is no possibility to restrain a brute force attack on the stolen key file performed on the machine of the attacker.

Resolving the situation becomes the more difficult, the more Trust Relationships to other Domains exist. Certificates of compromised keys, which are signed by a CA that belongs to a "classic" X.509 PKI, can be revoked, e.g. using revocation lists published by the CA that has signed the certificate. In our system this is not possible in case the compromised CA represents the root of a PKI hierarchy, i.e., if the compromised Domain CA is assigned to the home or company itself. In this case, no Domain CA exists "above" the compromised Domain CA, which could revoke the compromised certificate.

In this case the compromised Domain has no other option than contacting each Domain that trusts the compromised IK and convincing this Domain that the trusted IK got compromised. This, for instance, might be implemented using a signed data structure, a *Key Revocation Certificate*, created at the point in time the Domain CA is created. This approach is similar to the key revocation mechanism used by PGP/GPG.

But revoking the compromised Domain CA's certificate does not completely resolve the problem. All previously created Trust Relationships with other Domains need to be reestablished in order to introduce the new Domain CA's IK that replaces the compromised IK. For this purpose it might be possible to create a *Replacement IK* at the point in time a Domain CA is created. An additional data structure, an *IK Replacement Certificate* signed using the original IK might express that the certified *Replacement Key* may replace the compromised IK. If such an IK Replacement Certificate is not available, all Trust Relationships need to be renewed manually using the Trust Exchange process, which appears to be a cumbersome venture if Trust Relationships to numerous other Domains exist.

However, the attacker that stole the IK of a CA is easily able to create an IK Revocation and IK Replacement Certificate herself using the stolen IK. With these certificates she is able to cut off the compromised Domain from other Domains. This means that the attacker is able to hijack a complete Domain with all its Trust Relationships. In order to prevent this kind of attack, a fingerprint of the Replacement Key might be exchanged additionally during the Trust Exchange process between Domains. So once a Domain receives the IK Replacement Certificate at a later point in time, the Domain can compare the fingerprint with the public key of the new IK included in the IK Replacement Certificate. This approach will finally prevent the described Domain hijacking.

From the sides of other Domains that have a Trust Relationship with the compromised Domain, the effects of identity theft are fairly easy to resolve once the identity theft is detected. If the trusting Domain notices abuse of services, she simply has to remove the trusted certificate of the compromised Domain from her database, i.e., terminate the Trust Relationship. However, if the attacker is careful enough, the abuse might remain undetected for some time.

Theft of an Entity's Identity Key

Entities are devices or services that belong to a Domain. It appears to be much easier to steal the Identity Key from an entity than from a Domain CA, which resides within a shielded and well-protected Security Service VM hosted on the trustworthy Domain Server. This is especially true, if this entity is mobile, as it is quite easy for the attacker to obtain physical access. Additionally many devices are not well secured, which makes it easier for malware to compromise the system. As already discussed above, simple protection mechanisms, such as PIN/password protection for asymmetric key files, do not offer much additional security and are insufficient in this case as well.

Despite the smaller effort needed by the attacker to steal an entities IK, she gains almost the same "benefits" compared to stealing the IK of a Domain CA. The attacker is able to

access the same services, which means services offered in the compromised Domain and services shared by other Domains with the compromised Domain.

Luckily, resolving the impact of an identity theft of an entity's IK is less difficult compared to the theft of a Domain CA's IK. This is because the certificate of the stolen entity IK can be revoked by the Domain CA that has issued this certificate. Local or remote services are then able to check this Domain CA's certificate revocation list and refuse to accept a revoked certificate finally. The last step needed to resolve the IK theft is to deploy a new IK to the entity.

Synthesis:

Above considerations have shown that identity theft is a severe problem in both cases (Domain CA/entity). Although there exist various possibilities how the negative effects of identity theft can be resolved once the theft is detected (revocation/replacement of the compromised IK) this mitigation might come too late if the identity theft remains undetected for some time.

In theory, identity theft can be prevented quite easily. For this purpose a mechanism is needed that guarantees that a private key (the IK) can never be extracted from the Domain CA or entity.

13.1.2 Abuse of Identity Keys

The basic assumption of our hybrid trust model is that the owners of Partner Domains are trustworthy. Nevertheless we are aware that this might not always be the case. Additionally, it is possible that a formerly trustworthy Domain Owner turns rogue at some point in time later.

It is obvious that a "treacherous" Domain Owner can abuse the services or data shared by other Domains with her. We denote these Domains as *victims* in the following. But a treacherous Domain Owner can additionally assist third parties, which we call *accomplice*, to access services or data shared by a victim Domain. Therefore, attacks are subsequently analyzed where a treacherous Domain Owner assists her accomplice to attack victim Domains. The following analysis is performed from the perspective of the victims.

Passing the Identity Key of a Domain CA to an Accomplice

The first option the treacherous Domain Owner has to help her accomplice is to give the Identity Key of her own Domain CA to the accomplice. The negative effects on the security of the victim Domain are the same as if the accomplice would have stolen the IK.

Once the victim Domain has detected the attack, which appears to be difficult if the adversaries are careful enough, the problem can be fairly easy resolved. As previously described, the victim only needs to terminate the Trust Relationship with the treacherous Domain. Nevertheless, Trust Relationships between the treacherous Domain and *other* victim Domains still persist and can be abused as well.

From the viewpoint of victim Domains, this type of attack can be excluded if a Domain Owner is not in full control of the environment where the IK of her Domain CA resides. In this case, the treacherous Domain Owner is not able to extract the own Domain CA's IK and pass it to her accomplice. For this reason, the IK of a Domain CA must reside in an environment that only offers restrictive access to the IK. This means, that the usage of the IK to sign/encrypt or decrypt data is permitted, but the Domain Owner is not allowed to extract the IK.

Signing a Certificate for the Accomplice

The next option a treacherous Domain Owner has to help her accomplice is to sign a certificate for an entity owned by her accomplice. The effects on the security of the victim Domain are the same as if the accomplice would have stolen the IK of a device that belongs to the treacherous Domain.

Luckily, once this attack is detected, possible negative effects can be resolved easily by removing the treacherous friend's Domain Certificate from the database of trusted certificates. Unfortunately, it is principally not possible to prevent this attack as the Domain Owner is allowed to sign certificates of all entities she pleases. Hence, the treacherous Domain Owner is able to issue certificates to entities owned by herself or her accomplice as well.

In this context we want to point out that besides assigning the *same* access rights to *all* entities of a domain our access control system is also able to assign access rights to *individual* entities of a domain, see Chapter 7. This ability mitigates the effects of this attack as it is possible to assign no or only a small amount of rights to newly registered entities as these might be considered as particularly problematic in this context. Older entities might gain an increasing level of trust over time and obtain a growing amount of access rights. When such policies are used, the accomplice will only get very limited access, as she possesses a fresh and "untrusted" certificate. Unfortunately, the positive effect of this approach will be made futile when we consider the following attack.

Passing an Trusted Entity's Identity Key to the Accomplice

A treacherous Domain Owner might extract the Identity Key of one of her entities and send it to her accomplice. The effect is that the accomplice will gain access rights to services explicitly shared by the victim's Domain with this particular entity. This attack would render the above-described approach based on specific access rights for individual entities useless.

Though, it is principally possible to prevent this attack based on the abuse of IKs of trusted entities. Again, the environment that contains the IK must be restrictive enough and prevent that an IK can be extracted.

Synthesis:

The above-described attacks are especially displeasing as a trusted but treacherous Domain Owner is involved. As we discussed before, it is theoretically possible to prevent attacks where a treacherous friend gives a trusted IK to her accomplice. For this reason, a mechanism is needed that guarantees that the IK cannot leave the environment (Domain CA/entity) where it belongs to. Attacks that emerge after the treacherous friend signed a certificate for an entity that belongs to her accomplice cannot be excluded. As we have discussed, it is only possible to mitigate the negative effects that emerge from this attack when a modified access control policy is used by victim Domains that only gives limited access to newly registered and therefore less trustworthy entities. Nevertheless, this mechanism will only be effective, if such IKs are protected as described above.

13.2 Requirements

Based on the threat analysis performed above we define several requirements on a mechanism for key protection able to prevent the described attacks.

- **RF.1: Binding of an Identity Key to a Domain CA/Entity:** The Identity Key of a Domain CA, of a service or of a device must be inseparably bound to the environment the key should reside in. This will prevent identity theft by attackers or passing a trusted IK to accomplices.
- **RF.2: Provable Protection:** The protection of an Identity Key, i.e., this key's inseparable binding to the environment it belongs to, must be provable to other Domains. Without this provability, other Domains cannot have more confidence into the secrecy of an IK than they could have in an entirely unprotected IK.

Besides these most crucial requirements, various additional requirements exist that were already discussed in previous chapters.

- **RF.3: Resilience of the Identity Key:** A mechanism is needed that allows to backup the Identity Key in a secure and reliable manner in order to prevent against problems caused by the loss or destruction of this key.
- **RF.4: Usability:** The mechanism that provides the provable protection of the Identity Key may not impair the usability of the whole access control system. For instance, the migration of a Domain from one Domain Server to the other must still be possible.
- **RF.5: Low-Cost:** The key protection mechanism must be inexpensive.

Please note: Already at this point it is quite obvious that some Requirements are conflicting. A key that can be moved to another environment (RF.4) is naturally not inseparably bound to the environment it should reside in (RF.1). The protection mechanism for keys will therefore require a compromise between high security on one hand side and usability on the other hand side.

13.3 Discussion of the State of the Art

As already discussed, the overall idea to harden the access control system is to contain the Identity Key used by a Domain CA to sign certificates or by an entity for authentication purposes in some sort of restrictive/secure environment that guarantees that the key cannot be stolen or exported from this environment. As we have discussed, software-based approaches are prone to various (software-based) attacks and do not offer enough security. For this reason, the focus of the following considerations is on hardware-based systems for key protection and handling.

13.3.1 Cryptographic Token

Cryptographic token were introduced (see Section 9.2.1) and used (see Section 11.3.3) before. They are a specific type of smart card able to contain a private key in a secure environment. The private key is not only stored securely inside the card, but also does not need to leave the card when data needs to be signed/encrypted or decrypted. Cryptocards are also rather inexpensive (RF.5). Prices range from about five to fifteen Euros, which makes cryptographic token to a widely deployed, person-bound solution applied to many scenarios.

Cryptographic token are usually able to import an externally generated key pair, which might be generated in software on a computer. This can be regarded as a possible way to implement a mechanism for key resilience. For instance, an asymmetric key pair can

be generated on a secure computer, imported to the cryptographic token and additionally be backed up on some reliable storage medium (CD-ROM, USB memory stick, etc.), which needs to be kept in a safe place. With this described procedure, requirement RF.3, resilience, can be satisfied.

The final requirement RF.4, usability, is satisfied by cryptographic token partially. The needed migratability of a Domain CA's IK to another Domain Server is easy to achieve as the token can be moved easily. However, especially in combination with mobile devices used away on a journey, Cryptocards might be regarded as a burden by the user, as the card itself and an appropriate card reader need to be present all the time.

Cryptographic token do only partially fulfill requirement RF.1 as the card itself cannot be bound to a specific device. For this reason, the card and the included key can be stolen quite easily by an adversary that has physical access to the device or be passed by the Domain Owner easily to others. Another shortcoming of cryptographic token is that it is not possible to prove properties of a key to other parties, e.g., that the key is protected inside a Cryptocard (RF.2).

13.3.2 Hardware Security Modules

Hardware Security Modules (HSM) are cryptographic devices designed to guarantee the security of cryptographic keys and operations or to increase their speed. The range of HSM products is extensive and confusing. Properties of different products differ greatly. HSMs are typically certified according to the FIPS 140-2 security standards, which describe different security requirements for cryptographic modules. Depending on processing performance offered and compliance to security levels, prices range between about fifty and several thousands of Euros, which clearly contradicts RF.5. Therefore, HSMs are typically deployed in professional environments such as commercial Certificate Authorities, the online banking sector or domain name registrars (DNSSec).

The only common property HSMs seem to have is their ability to contain one or more private keys and to use these keys inside an own, isolated processing facility for cryptographic operations. Therefore, HSMs are expected to provide a high level of security against attacks performed in software. Typically HSMs are hardware devices that are either built into a computer (PCI card) or plugged into the computer (USB device). In both cases the HSM and the key contained in it are not inseparably bound to the device. Therefore, we regard RF.1 as only partially fulfilled.

For the remaining requirements (RF.2, RF.3, RF.4) no definitive answer can be given due to the wide range of existing products. However, the Author is unaware of an HSM type able to prove the protection of a key contained within the HSM's isolated environment to another party.

13.3.3 The Trusted Platform Module

The *Trusted Platform Module* (TPM) introduced in Section 9.3.2 and used as trust anchor for integrity measurements in Chapter 12, is a cryptographic chip designed for being integrated into main boards of commodity computer systems. As the chip is inexpensive (RF.5), TPMs are often included into off-the-shelf computing systems (notebooks, desktop PC, server) per default.

The TPM is capable to protect asymmetric keys inside its shielded environment and to use these keys for cryptographic operations therein. The TPM is also able to guarantee that certain types of private keys cannot leave it. Finally, the TPM itself is bound to a device, as the TPM chip is inseparably soldered to this computer's main board. By these reasons is RF.1 completely fulfilled by the TPM.

The TPM is integrated in a far-reaching ecosystem of TCG standardized technologies, which offer interesting opportunities. One main feature of the TPM is its ability to prove properties of keys to a party that wants to use this key. This proof includes, for instance, that a specific private key cannot leave the TPM. By these reasons the TPM fulfills Requirement RF.2.

However, many beneficial properties of the TPM are based on a very restrictive interface. Due to these restrictions, e.g., the guaranteed inseparable binding of a key to the TPM, it must be expected that the usage of the TPM technology creates various problems concerning key resilience (RF.3) and usability (RF.4).

13.3.4 Synthesis

Despite the mentioned problems with Trusted Computing technology, the TPM is the most promising key protection mechanism, as it is the only technology that fulfills the most important requirements (RF.1, RF.2) we have defined. Therefore, Trusted Computing Technology is explored further in the following in order to understand how a concept to use the TPM as safeguard for identity keys can be created.

13.4 Design

13.4.1 TPM Integration Concept

The integration of the Trusted Platform Module into the already described access control system as safeguard for Identity Keys seems straightforward at first sight. But, TCG specifications of the interface to the TPM and the properties of different TPM key types must be well understood in order to achieve the best level of fulfillment of requirements, which were defined above.

13.4.1.1 TPM Signing Keys as Identity Keys

The first question that must be answered is which TPM key type is suitable as Identity Key for a Domain CA or entity. For this purpose, a TPM key type is needed that is guaranteed not to leave the TPM but also able to create cryptographic signatures of data that originates from the *outside* of the TPM. Examples include signatures of certificates of other IKs (Domain CA) or signatures of challenges needed as a response in cryptographic authentication protocols (entities). Finally, the protection of an IK through the TPM must be provable.

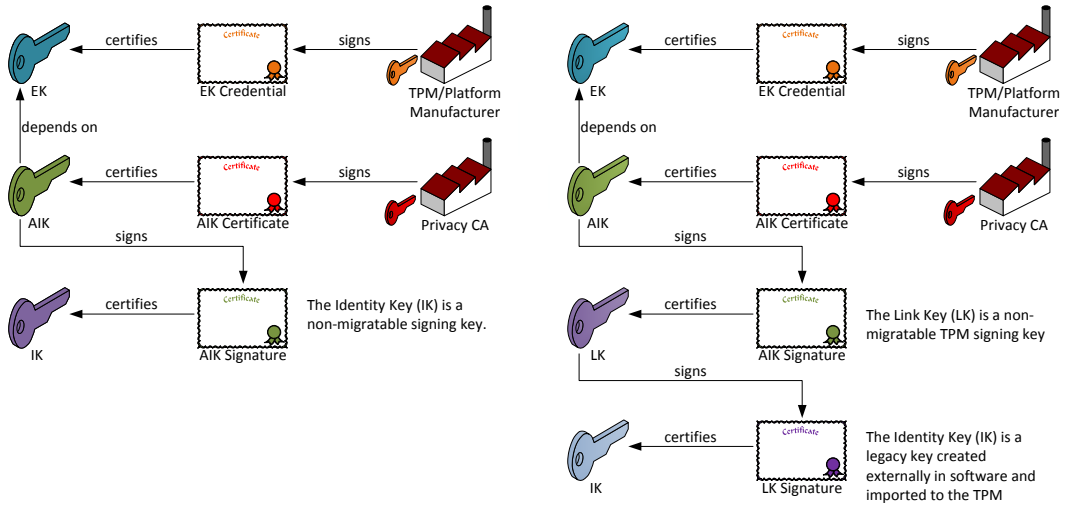
The only TPM key type that meets these requirements is a non-migratable *Signing Key*. Other TPM key types, which may sign data, namely the EK (Endorsement Key) and AIKs (Attested Identity Key) cannot be used for the intended purpose, as they are only allowed to sign so called *TPM_CERTIFY_INFO* data structures of other non-migratable TPM keys (EK, AIK) and TPM_QUOTE_INFO data structures that contain contents of the TPM's PCRs (AIK).

The non-migratability of an IK of TPM key type Signing Key can be proven to others using the just mentioned TPM_CERTIFY_INFO data structure. This data structure is generated and signed within the TPM using an AIK. It contains a reference to the public part of the IK and evidence of the IK's properties, i.e., it proves that this specific IK is non-migratable. In the following we denote this data structure simply as *AIK signature*. The IK owner can later prove the TPM protection of her IK to others by presenting AIK Credential and AIK signature to the party that wants to validate the properties of the IK.

The AIK and the AIK credential, which are needed for this validation, need to be generated by the TPM resp. obtained from a Privacy CA, which exists outside the local network.

For this purpose, the Privacy CA validates the Credentials of the Trusted Computing Platform, see Section 9.3.2.3, and a TPM_CERTIFY_INFO data structure signed by the EK, which gives evidence about the AIK's properties.

The described process results in the short key and certificate/signature chain depicted in Figure 13.1(a).



(a) IK of Type Signing Key.

(b) IK of Type Legacy Key.

Figure 13.1: Certificate/signature chains for different IK types.

One important restriction exists when a non-migratable Signing Key is used. The TPM refuses to export the private part of this key, which prevents that Requirement R3, namely key resilience, and R4, namely usability (migratability), can be fulfilled.

13.4.1.2 Key Resilience and Usability

As we have explained above, a non-migratable Signing Key used as IK cannot be backed up or migrated to another device as the TPM refuses to export this key. The TCG has recognized this problem and has standardized two mechanisms that involve trusted intermediate authorities and a new TPM key type. In the following we analyze their applicability to our scenario.

Maintenance

The first mechanism is called *TPM maintenance* [71, p. 112ff] and provides a resilience mechanism for TPM protected keys. Maintenance is a process that involves the TPM owner, who is the TPM super user, and the platform manufacturer. Simply put, TPM maintenance generates an encrypted binary file containing all keys protected by the TPM, which can be stored at some safe place. Neither the TPM owner nor the platform manufacturer is able to decrypt this file on her own. In the case that the original TPM or platform become malfunctional or are destroyed, the TPM owner may buy a new device and import the encrypted binary file with the help of the platform manufacturer into it.

TPM maintenance is a valid concept but it has some major drawbacks. First, maintenance is an *optional* feature of the TPM, i.e., it is possible that the used TPM is not able to perform TPM maintenance at all. Second, if the TPM is used by multiple users, e.g., by multiple Domains hosted on the Domain Server, all users must trust the TPM owner to create the necessary backup, as an ordinary TPM user is not allowed to perform TPM maintenance.

Certified Migration

Another concept, which becomes interesting for moving a Domain (migrating an IK) to another Domain Server, was introduced in version 1.2 of the TPM specification. The introduced *Certified Migratable Keys (CMK)* [71, p. 85ff] can be migrated by ordinary TPM users, in our case a Domain Owner, between TPMs with the help of an intermediate authority called *Migration Authority (MA)*. Without the help of a MA, a CMK cannot be migrated and therefore has properties comparable to an ordinary, non-migratable key. The MA and specific migration protocols, which resemble the TPM maintenance protocols, guarantee that the key in migration can neither be exposed to the destination nor the source platform nor to the MA itself, i.e., that the key is securely transferred from one TPM to the other.

Again, the described concept is valid. Nevertheless the question must be answered which MA can be used. To the Author's knowledge, no publicly available MA exists at this point in time. For this reason, the Domain Server Manufacture would have to provide a MA for Domain Servers.

Synthesis

The official TCG procedures for TPM maintenance and certified migration are valid. Nevertheless, when used in practice, several questions remain open. Therefore, a different, more usable variant of the previously described TPM integration concept is presented in the following that does not involve any intermediate TCG authorities.

13.4.1.3 Legacy Keys as Identity Keys

The only option how an IK can be migrated to another TPM or be backed up without the support of the previously described intermediate TCG authorities is if the IK is a TPM Legacy Key. A Legacy Key is a key, which is generated in software (OpenSSL) outside the TPM and later imported into the TPM. Naturally, such a key can be copied to a backup medium before it is imported into the TPM. Furthermore, it can be transported (migrated) to every desired location where it can be imported into a TPM. The resulting modified certificate/signature chain is depicted in Figure 13.1(b).

The first two layers of the shown certificate/signature chain are equivalent to the corresponding layers of the chain already explained in Section 13.4.1.1. On layer three of the modified chain a first modification can be observed. Here, the so-called *Link Key* is introduced. This key is a TPM Signing Key, which is used to "link" the IK to the TPM it is imported into. This link is established by a signature created by the Link Key of the public part of an imported IK. This so called *LK Signature* expresses that the IK is a migratable Legacy Key that was imported into a specific TPM.

The proposed variant trades some aspects of key security, namely the inseparable binding provided by an IK of type Signing Key, for the sake of flexibility and resilience provided by an IK of type Legacy Key.

This drawback is quite unimportant for the Domain Owner herself, as the IK, once it is imported into the TPM, can only be exported by the Domain Owner after providing a specific secret. The drawback of a Legacy Key used as IK is more important for other Domains. The guarantee that the key owner cannot abuse this key by passing it to her accomplice is lost. However, the described modified certificate/signature chain still proves the properties of the IK, namely that the IK cannot be stolen from the TPM by an adversary.

13.4.2 Protocol Flows

After explaining which options exist to integrate the TPM as safeguard for identity keys into the existing system, new protocols must be introduced that are needed to initialize the described key and certificate/signature chains. Moreover, the Device Registration protocol needs to be modified.

13.4.2.1 TPM Initialization

Instead of simply generating an asymmetric key pair in software later used by the Domain CA or an entity, the TPM must be initialized and the key and certificate/signature chain must be generated.

IK of Type Signing Key

Figure 13.2 displays the steps required if the user decides to utilize a Signing Key as IK. The result of this protocol is the certificate/signature chain depicted in Figure 13.1(a).

In Step 1 the user decides to put the TPM of her TCP into operation and to generate an IK of type Signing Key. This decision is transferred to the *TPM Manager*, which is the software component we developed that encapsulates all necessary functionality to deal with the TPM. Besides the TPM and key initialization described here, the TPM Manager is used by the Registration Service and the Registration Client during device registration, see Section 13.4.2.2.

Steps 2 - 4 are needed to take ownership of the TPM. This procedure only needs to be performed once, for instance by the owner of the Domain Server when she generates the Domain CA for the local network. The user that has taken ownership of the TPM will become the super user of the TPM. The TPM Manager will omit this step, when other stakeholders of the Domain Server later generate a Domain CA, e.g., for their User Domain. After taking ownership, a new AIK is generated in Steps 5 - 7. The necessary AIK Credential is received from a Privacy CA that exists outside the local network in Steps 8 - 15.

The Steps 16 - 18 generate the IK of type Signing Key. Subsequently the AIK Signature of the public part of the IK is computed in Steps 19 - 22. With this step the initialization of the key and certificate/signature chain for the IK is finished and the IK is ready to be used.

IK of Type Legacy Key

If the user decides to utilize an IK of type Legacy Key, the key and certificate/signature chain as displayed in Figure 13.1(b) needs to be generated. For this reason, the previously explained protocol must be modified. The first steps of the modified protocol are the same as Steps 1 - 15 of the protocol depicted in Figure 13.2. For this reason these steps are not repeated here.

The first deviation found in the modified initialization protocol, see Figure 13.1(b), are Steps 16 - 18. Here the Link Key is generated, which is subsequently signed by the AIK (Steps 19 - 22).

Now the IK is generated in software, e.g., OpenSSL, outside the TPM (Step 23). Before the key is imported into the TPM, it will be stored on some backup medium, see Steps 24 - 26. To guarantee the secrecy and resilience of a Domain CA's IK, the same mechanism as described in Chapter 11 can be used. Next, the IK is imported into the TPM (Steps 27 - 29) and the LK signature of the public part of the IK is computed in Steps 30 - 33. With this step the initialization of the key and certificate/signature chain for the IK is finished and the IK is ready to be used.

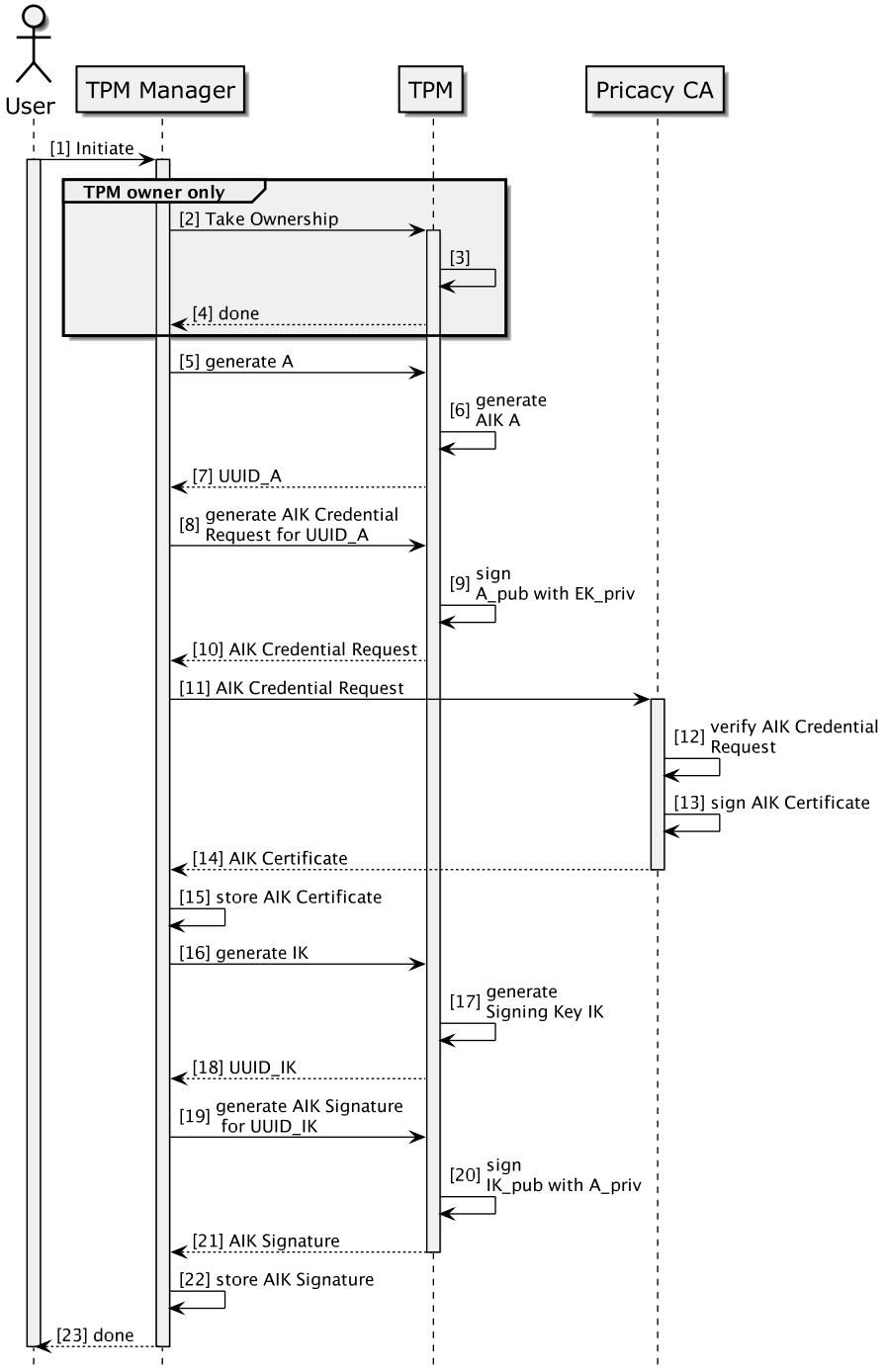


Figure 13.2: Initialization protocol of the key and certificate/signature chain for an IK of type Signing Key, see Figure 13.1(a).

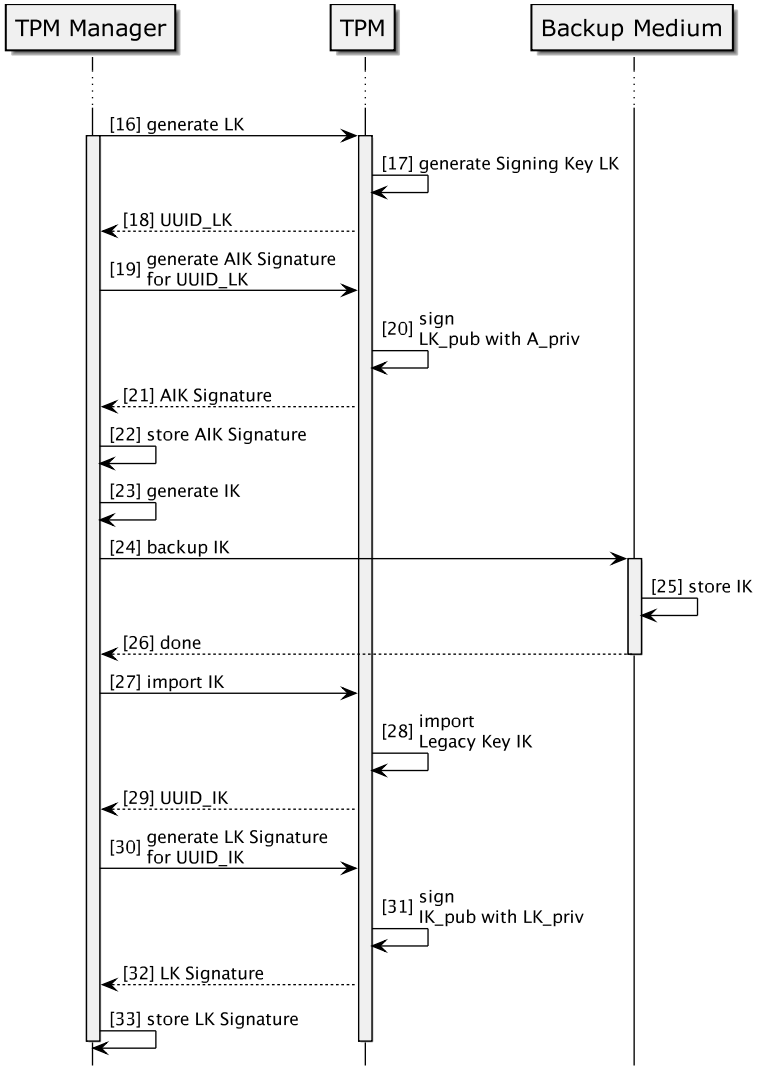


Figure 13.3: Initialization of the Key Hierarchy with an externally generated Identity Key (Legacy Key), see Figure 13.1(b). Steps 1 - 15 (not depicted) are equivalent to Figure 13.2.

13.4.2.2 TPM-aware Device Registration Protocol

Figure 13.4 depicts an extension of the original Device Registration Protocol, which was explained in Section 5.3. The extensions replace Steps 26 - 31 of the original protocol in order to make the certification TPM-aware, i.e. to determine the key type of the IK that must be certified and to create the signature using the Domain CA's IK contained inside the TPM of the Domain Server.

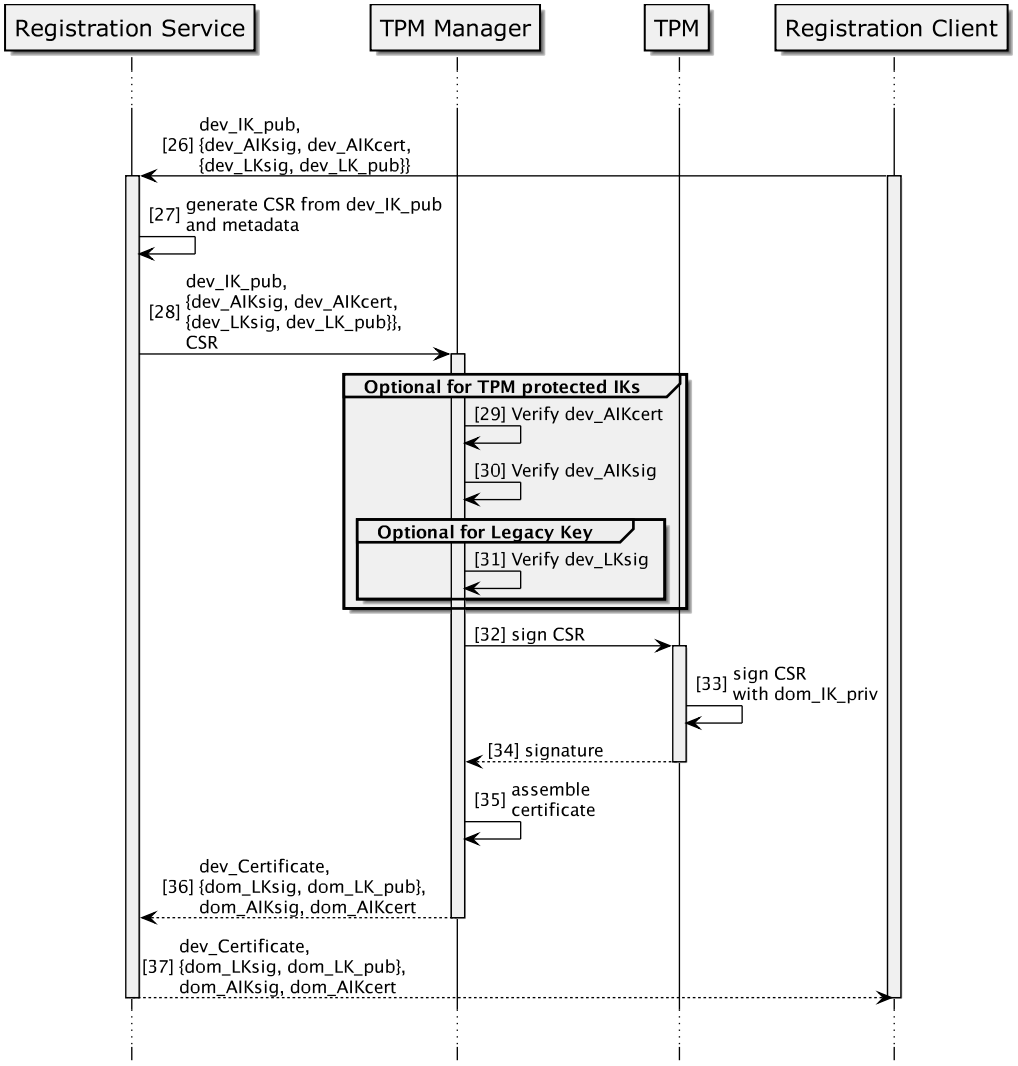


Figure 13.4: The TPM-enabled Registration Service issues a Device Certificate after evaluating the properties of the devices IK. The depicted steps are performed instead of Steps 26 - 31 shown in Figure 5.3.

The present figure and explanations introduce actions at the side of the Registration Service. These steps are needed to check key properties of IK of the registering device and to sign the certificate for this IK using the Domain CA's TPM-protected IK. The described protocol requires that both, the registering device and the Registration Service, are TPM-enabled and have already initialized their personal IK as described before (see Section 13.4.2).

In Step 26 of the modified Device Registration protocol, the Registration Client installed on the registering device sends the public part of its IK (dev_IK_pub) to the Registration Service installed on the Domain Server. If this IK is TPM protected, the certificate/signature chain and all public keys needed to validate the presented certificate/signature chain need to be sent additionally.

As in the original protocol, the Registration Service generates a Certificate Signing Request (CSR) for dev_IK_pub (Step 27). This CSR is then transferred together with all credentials received previously from the Registration Client to the TPM Manager (Step 28).

Now the TPM Manager starts to verify the received credentials in order to establish trust in the IK that needs to be certified. Depending on the settings of the TPM Manager it is possible to restrict it to only sign IKs of type Signing Key, both IK types (Signing Key/Legacy Key), or all IK types, no matter if TPM-protected or not. The TPM manager can easily determine the properties of the IK by evaluating the received certificate/signature chain.

For this purpose, the TPM Manager first verifies the AIK Credential (dev_AIKcert), if present, using the certificate of the Privacy CA that has signed the credential in Step 29. If the TPM Manager is convinced that the used AIK is trustworthy, it will use the corresponding public key (dev_AIK_pub) to validate the AIK Signature (dev_AIKsig) in Step 30.

If the IK is a non-migratable Signing Key, the validation of the certificate/signature chain is already finished and the TPM Manager may sign the CSR. If the registering devices IK is a Legacy Key, then the optional Step 31 needs to be executed. Here the TPM Manager checks if the LK signature of the public key of the IK is valid.

After the TPM Manager has verified the presented certificate/signature chain, the CSR is signed by the TPM using the private part of the Domain's IK (dom_IK_priv) in Steps 32 - 34. Finally, the certificate is assembled by the TPM Manager using the just computed signature. After this step, the TPM-signed device certificate and the certificate/signature chain of the Domain CA's IK are sent to the Registration Service (Step 36).

The Registration Service finally forwards all data in Step 37 to the Registration Client, who stores the received data for later authentication purposes.

13.4.3 TPM Integration into the Virtualized Domain Server

The initial specifications of the TPM and the TCG ecosystem date back to 2003 and earlier. At this point in time hardware virtualization was not as widely deployed as it is today. Therefore, virtualization support did not play an important role in the specifications and TPM integration to virtualized systems is today still solved improperly.

With the widespread deployment of virtualization the research community introduced various approaches that provide TPM functionality to VMs. One proposed option is to emulate a TPM in a VM entirely in software. This reduces the idea of the TPM being a hardware trust anchor to absurdity. Hence, this approach was not used in this work. Another approach, as described in the research paper by Berger et. al. [91], is to multiplex the TPM of the physical machine. The approach uses a specific driver architecture that presents an own, *virtual TPM (vTPM)* to each VM. TPM commands invoked by an application that resides in a VM are transported by the vTPM frontend drivers via the hypervisor to the vTPM backend driver who multiplexes TPM access and finally sends the command to the physical TPM. In 2011 a TCG specification influenced by this research work was published [92]. Our experiments with an early implementation of the proposed technology for the Xen virtualization system conducted in 2009/2010 revealed various problems. For instance, a reboot of the TCP caused the loss of keys generated by all vTPMs.

To allow the different Registration Services hosted in individual Security Service VMs to use the physical TPM of the Domain Server a work around had to be used. The *Java Trusted Software Stack (jTSS)* [93], which is a specific implementation of the TCG standardized interface to the TPM, offers besides local TPM function calls a SOAP interface.

This SOAP interface can also be accessed from an application over the network, for instance, a TPM manager running within a Security Service VM. So every time the TPM Manager must perform an operation that requires the physical TPM, a remote procedure call is sent to the jTSS located in the dom0 of the Domain Server.

13.5 Discussion and Evaluation

13.5.1 Fulfillment of Requirements

This chapter proposed the usage of the Trusted Platform Module as hardware safeguard for Identity Keys for Domain CAs and entities. More specifically, two different TPM key types used as IK were proposed. Both key types result in different properties of the offered key protection.

The first IK type proposed is a TPM Signing Key. Such a key is inseparably bound to the TPM and guaranteed to never leave it (RF.1). Additionally, this binding is provable (RF.2) to other parties using the first discussed certificate/signature chain, see Figure 13.1(a). A Signing Key used as IK offers the highest amount of security. But, if no TCG intermediate authorities for TPM maintenance and certified migration are available, the usability and resilience of the IK (RF.3/RF.4) are severely impaired. In fact, these drawbacks might be even greater than the gained security benefit.

The second TPM key type that can be used as IK is a Legacy Key, which is a software-generated key imported into the TPM. The major benefit of using a Legacy Key as IK is that this IK does not limit the usability (RF.4), e.g., concerning Domain migration to another Domain Server, and that it is easily possible to create a key resilience mechanism (RF.3). A Legacy Key is not inseparably bound to a TPM but the TPM still protects this key, which means that the IK cannot be extracted from the TPM by an adversary. But it is possible that the key owner gives the key to somebody else who might abuse it. For this reasons RF.1 is only fulfilled partially. As we have discussed, the described properties of the key can be proven to the party that wants to use the IK by presenting the modified certificate/signature chain depicted in Figure 13.1(b) (RF.2).

The following Table 13.1 compares properties of an unprotected IK (Software) as used so far by Domain CAs or entities, of the two TPM-protected IK types (Signing or Legacy Key), and the standardized solution by TCG that uses Certified Migratable Key as IK. The last option can only be used when TCG intermediate authorities for TPM maintenance and certified migration are available. The first column of the table contains a property of the key protection mechanisms. The remaining columns answer the question if this property is fulfilled by an IK.

			This Thesis	
	Software	TCG	**Signing Key**	**Legacy Key**
RF.1: Prevent theft	✗	✓	✓	✓
RF.1: Prevent passing	✗	✓	✓	✗
RF.1: Prevent signing	✗	✗	✗	✗
RF.2: Provability	✗	✓	✓	✓
RF.3: Resilience	✓	❍[2]	✗	✓
RF.4: Usability	✓	❍[3]	✗	✓

Table 13.1: Comparison of the state of the art and the solution of this Thesis. [2]Due to unavailable TPM Maintenance functionalities. [3]Due to unavailable Migration Authorities.

13.5.2 Recommendation

If the TCG standardized intermediate authorities are not available, an IK of type Legacy Key should be used for a Domain CA. On first sight this recommendation is contra-intuitive as the Legacy Key offers less protection than a non-migratable Signing Key. The reason for this recommendation is that the Domain CA's IK must be seen as a long-term key that must be highly resilient as this key is used for many years. Also the ability to migrate the key to another Domain Server is highly important in the case that the Domain Owner moves. The described partial loss of key security does not affect the Domain the key belongs to. Only those Domains that trust into this IK lose some protection because the Domain Owner can give his IK to somebody else. But, as discussed, a treacherous Domain Owner can also issue a certificate to her accomplice, which principally cannot be prevented by any key protection mechanism.

For entities an IK of TPM key type Signing Key should be preferred. Here resilience and usability/migratability do not play such a big role, as an entities key can be easily replaced. Additionally, entities are more likely to be exposed to adversaries than a Domain CA located on the secure Domain Server. For this reason the key type that offers most security should be selected.

13.5.3 Limitations

Besides the limitations of Trusted Computing Technology already described in Section 12.6, further limitations were found.

At the beginning of this work we researched if the TCG intermediate authorities for TPM Maintenance and Certified Migration are publicly available. By the knowledge of the Author these authorities do not exist publicly.

We also discussed the limitations of Trusted Computing in combination with virtualization. Future work, which might also be influenced by the upcoming version 2 of the TPM, might focus on the integration of vTPMs into existing hardware virtualization systems.

13.5.4 Future Work

A domain that establishes a Trust Relationship to another domain can derive from the presented certificate/signature chain of the Partner Domain CA the degree of IK protection (unprotected key/Signing Key/Legacy Key). Mechanisms that extend the Trust Exchange and access control mechanisms described in Chapters 6 and 7 with this knowledge might be part of future research. For instance, depending on IK protection and policy of the Domain, the Trust Exchange mechanisms might refuse to accept a Trust Relationship to a Domain that does not protect its Domain CA's IK at all. Alternatively, only a limited set of default access rights might be assigned to such Domains. Vice versa, it is possible to equip especially protected devices with more access rights than unprotected devices.

13.6 Conclusion

One important problem of every access control system that uses asymmetric cryptography for authentication, such as the system proposed in this thesis, is the secrecy of keys. We first discussed that keys can get stolen by an adversary. But keys can also be abused by their owners, which we actually regarded as trustworthy. Such "treacherous" key owners might pass a trusted authentication key to somebody else in order to give this person access to another Domain's services.

We investigated several hardware technologies for key protection, namely cryptographic token, Hardware Security Modules, and the Trusted Platform Module. These technologies

all are able to offer a certain amount of protection to asymmetric keys. However, the Trusted Platform Module is the only technology able to inseparably bind a key to a platform, prevent some types of key abuse, and to additionally prove this protection to others. Based on this technology we described a first concept to leverage the TPM as hardware key safeguard for Domain CAs and entities.

The inseparable binding of a key to a TPM adds security but reduces usability and makes key resilience mechanisms suitable for unmanaged environments impossible. By these reasons a second concept to integrate the TPM into our system using a different TPM key type was proposed. The second concept reduces the security of the key protection slightly, but provides the ability to backup and migrate the TPM-protected key.

The proposed key protection mechanism can be integrated into Domain CAs, client devices or devices that host services. Domains that bind their trust and access rights to keys can benefit from this technology as well, as they can determine the trustworthiness of a certain key.

Key Findings and Contributions

▷	Theft or abuse of a private key are of major concern for every access control system based on asymmetric cryptography.
▷	The Trusted Platform Module, a cryptographic chip, is a suitable technology that offers protection against key theft and abuse.
C13.1	**TPM integration concepts to Domain CAs and entities**. The first elaborated option binds a key inseparably to the Domain CA or entity and offers most security advances. The drawback of this option is that keys cannot be migrated or backed up. The second option trades a small amount of security advances for flexibility, i.e., the ability to migrate and backup a key.
C13.2	The **TPM manager** implements these concepts and integrates TPM functionality to the Registration Service presented earlier.

Part IV

Conclusion

14. Conclusions and Future Directions

Security is one of the most important requirements on networks. For this reason a plethora of technologies have been developed in order to achieve network security. Access control is one of the most basic but also most important building blocks for network security. A common problem of many security technologies is that they are highly complex and can only be used in environments managed by professionals.

However, many examples for unmanaged networks operated by laymen exist, such as home networks. Modern homes have been evolving into highly complex networked environments, the so-called smart home, and have therefore high security demands nowadays. As shown in Sections 2.3 and 7.7.3 no serious security technologies targeted to homes exist today, and professional solutions cannot be deployed to this environment due to the missing experience of users.

For this reasons, we defined two central goals for this thesis: 1) to create an access control system targeted to unmanaged networks and 2) to create a device suitable to host and protect the components of this system. Our central claim was that based on existing enterprise grade security technologies and carefully designed service components that assist inexperienced users, a similar level of security (concerning access control) can be achieved in unmanaged networks as in professionally managed networks.

14.1 Central Findings and Contributions

In Section 1.1 we defined six research questions. In the following we sum up how we answered these questions and present our central findings and contributions. For a convenient overview and a comparison of our contributions to the state of the art and related work please refer to Table 14.1.

Q1

The first step needed to answer question 1, **how to design a user-centric identification/authentication scheme for unmanaged networks based on strong cryptography**, was to understand the properties of unmanaged networks. We used the home network as a central example to analyze technical and social properties and presented our findings in **Chapter 2**. The most important findings of the analysis were that 1) access control should not be managed by a single authority in the home network typically. Instead, the different residents of the home should manage their personal devices and

services themselves. Therefore, a home access control system must be highly distributed and user-centered. 2) Strong social relationships and the desire to share services and data exist between residents of a home. Both factors need to be taken into account by an access control system targeted to this environment.

In **Chapter 4** we developed the idea to partition the home network into several distinct **User Domains**. Each Domain is assigned to a resident and contains devices and services owned by this person. An individual **User Domain CA** represents a User Domain and issues valid X.509 certificates to devices and services that are members to this Domain. The home itself must be represented by a Domain as well. The *Home Domain CA*, which is assigned to this Domain, acts as root for the home's private public key infrastructure (PKI).

The resulting CA hierarchy or certificate chain of Home Domain to User Domains to entities (Home Domain Certificate → User Domain Certificate → Entity Certificate) is the basis for a **hierarchic and self-certifying identifier scheme**. The locally valid certificates or the certificate chain enables **secure authentication**.

The major advantage of this identification/authentication scheme is that it is entirely user-centered and does not depend on external PKIs. Problems concerning privacy, as with the PGP/GPG Web-of-Trust, or trustworthiness of external certificate authorities, as with public X.509 PKIs, c.f. Section 3.3, do not need to be feared.

Q2

The identification/authentication scheme presented so far creates two problems: 1) it is based on asymmetric cryptography and it requires the certification of keys. Asymmetric cryptography is difficult to understand and the certification process of a key is too difficult to be performed by laymen. 2) The certificates are only valid within the home, i.e., it is impossible to authenticate entities that belong to a different home.

To address the first problem we proposed in **Chapter 5** the simple to understand **Registration metaphor**. An entity must be registered to a Domain in order to become a valid member of the home and to be able to authenticate to services offered in this home. The **Registration Service** we designed and implemented is a service that distributes certified asymmetric keys to entities in a secure and user-friendly manner. The users of this service are guided through the process and do neither need to understand nor care about technical details of the certification process.

The second problem could be addressed by an approach we called *Trust Exchange*, see **Chapter 6**. The basic idea is to exchange Domain CA certificates between Partner Domains that want to share services with each other. The certificate exchange will finally enable both Domains to mutually authenticate each other. The first implementation of this approach follows a Direct Trust Model. This means that the key exchange partner is personally identified and the certificate exchange is performed directly between both persons. The users involved in this process are guided by a semi-automated mechanism called **Personal Trust Exchange** service. This service takes care that certificates are exchanged securely and finally deployed in the own Domain. This service is highly secure, but can only be used when representatives of both Domains are able to meet each other. The second approach is based on the Web-of-Trust Trust Model. The **Internet Trust Exchange** service exchanges certificates between Partner Domains over the Internet with the help of other mutually trusted Counselor Domains. This process is secure, provides strong identification between the Domains, and preserves the privacy of all involved Domains. The strength of a Trust Relationship established over the Internet can be rated using the **Trust Metric** we proposed.

With these contributions we were able to answer question 2, **how to empower inexperienced users to effortlessly and securely use this identification/authentication scheme**.

Q3

Being able to authenticate entities belonging to Domains of the own or another home based on strong cryptography is a great security advance. Nevertheless, access control solely based on authentication offers only coarse granularity and limited flexibility. For this reason we connected an authorization system to our authentication system that enables fine-grained authorization. A suitable authorization system is defined by OASIS in the XACML standard. The standard consists of a highly expressive authorization language and infrastructural authorization services. Due to the complexity of the XACML language we had to answer question 3, **how to empower inexperienced users to use a sophisticated authorization system**.

In **Chapter 7** we connected XACML technology to our identification/authentication scheme. For this purpose two technologies had to be developed: 1) **TLS Handshake Interception**, which extends TLS-based authentication protocols with the ability to perform XACML-based authorization additionally. The technology can be integrated easily into legacy applications that use TLS-based authentication. 2) A **Guided Policy Administration Point** that allows laymen to effortlessly specify authorization rules for their own Domain based on an easy to understand graphical representation of "known" Domains and available services. This system finally translates settings into a valid XACML Policy Set, which is used by the XACML authorization service.

Q4

The service components of our so-called **Guided Security Management System** need to be hosted on a secure computing device in the home. From the mostly inexperienced users in homes we cannot expect that they setup and maintain this device or the service components of the Guided Security Management System, take care for the device's security and back up the state of their Domain (the CA's key, access control settings, exchanged certificates, etc.). For this reason question 4, **How to design a device able to host security services in a home environment securely, resiliently and user-friendly**, had to be addressed.

In **Chapter 8** we analyzed and defined requirements on the architecture of the so-called **Domain Server** and Auxiliary Services that allow inexperienced users to conveniently achieve above described goals.

Chapter 10 described the basic architecture of the Domain Server based on the XEN hardware virtualization system. We leveraged XEN virtual machines (VMs) to isolate 1) security relevant service components of different Domains and 2) other services of different Domains from each other. This architecture is highly flexible, prevents various security related problems, and finally is the basis for other technologies we introduced later to increase the security of the Domain Server. Additionally we used XEN networking capabilities to provide and enforce the network topology needed by the Registration Service.

Based on the Domain Server architecture we designed **Auxiliary Services** that automate Domain management in **Chapter 11**. Our central idea was to facilitate Domain management using template VM images (**virtual appliances**) that contain a Domain's Guided Security Management System and other services related to security. For each Domain, an

individual VM is booted from this virtual appliance and later initialized with the Domain's state.

For automated Domain management we designed and implemented the **Domain Server Manager**. The **Resilience Service** we created performs the initialization of virtual machines with Domain-specific information. This service encrypts Domain state and uploads this information to an arbitrary networked storage service in the Internet. When a newly started virtual machine needs to be initialized, the service downloads and decrypts this information.

In order to interact with the Domain Server Manager and to store the encryption key used by the Resilience Service, we used cryptographic token cards. If a Domain Owner's card is plugged into the Domain Server, the Domain Server Manager starts the virtual machine containing all services of a Domain. If the card is unplugged, the VM is shut down. This solution is highly convenient and intuitive to use and requires no technical skills from users.

Q5

The Domain Server and the virtual appliances containing security related services must have integrity. This basically means that Domain Server and VMs only execute authorized and authentic software components. Otherwise the Domain Server, virtual appliances and finally the Guided Security Management Systems cannot be trusted, and various security problems might emerge. In order to answer question 5, **how to protect and guarantee the integrity of this device**, we created a unique combination of two enterprise grade security technologies, namely the Trusted Platform Module (TPM) and Java smart cards.

In **Chapter 12** we described how the TPM can be used for integrity measurements of the Domain Server. A problem we encountered is that the state of the art does not offer a suitable mechanism that assesses the integrity of the Domain Server based on TPM-protected integrity measurement values for our scenario. For this reason, we designed and implemented an **Integrity Verifier applet** for Java smart cards, which verifies the Domain Server's integrity once the card is plugged in. Additionally, we extended the Domain Server Manager with a functionality that only starts virtual appliances after their integrity could be proven.

The combination of both mechanisms establishes a chain of trust from Java smart card to Domain Server to virtual machine. So when the Resilience Service located in a virtual machine requests the Java smart card to decrypt the Domain's state, the smart card will only perform this operation when the Domain Server has integrity. This innovative combination of our mechanisms effectively prevents that personal information (e.g., Domain state) is revealed to an untrustworthy, maybe compromised Domain Server.

Q6

The final question 6, **how to protect keying material used by the identification/authentication scheme against theft and abuse**, could be addressed using the TPM as well. The TPM is a highly useful technology in this context, as it is the only technology we are aware of able to protect keys against theft and abuse, even by their owners, and to prove the protection of a key to others in a trustworthy manner.

In **Chapter 13** we discussed **how the TPM can be conceptually integrated** into our system as hardware key safeguard. The first option we proposed for TPM integration binds a key inseparable to the TPM of Domain Server or entity. Therefore, key theft and various types of abuse, even by the key owner, can be prevented. Unfortunately,

this option is problematic considering key resilience. Although the Trusted Computing standards describe mechanisms that allow to backup TPM keys they are not practicable due to missing TPM or device manufacturer support. For this reason we proposed a different TPM integration concept. This second option trades the TPM's ability to prevent key abuse by the key owner with the ability to backup the key.

We implemented these concepts in the **TPM manager** and integrated this software component into the Registration Service. With this enhancement, a Domain CA whose key is protected by a TPM can certify keys of devices, which can also be protected by a TPM. This results in highly trustworthy certified authentication keys that are not prone to identity theft and some types of abuse by their owners, which finally strengthen the Trust Relationships between Domains.

At this point we finally want to emphasize that the contributions of this thesis are not only applicable to the home environment. The reason why our technologies are also applicable to managed, professional environments is their foundation on standardized technologies that are used in enterprise environments today. Therefore, our technologies may act either as a replacement of a system administrator in smaller company network or to disburden the administrator.

14.2 Overview on the Access Control System

In Figure 14.1 we give an overview on the access control system presented in this thesis. The figure shows two different home networks with their Home Domain's Security VM (green block) hosted on the Domain Server (see Chapter 10), a device (orange block), and a service (blue block) each.

In the Security VM all service components of the Guided Security Management System are located, namely the Registration Server, the Trust Exchange Server and the Guided Policy Administration Point. Furthermore, XACML-related services, such as the PDP and Proxy PEP, are executed in this VM. The Registration Client and the application for Personal Trust Exchange are installed on devices. Additionally, devices are equipped with client applications for networked services provided in the home. Services can be either hosted on a virtual machine running on the Domain Server or on another device located in the home.

Before a device can be used in the home network it must be registered ① (see Chapter 5). As a side effect the device is equipped with a certificate signed by the local Domain CA. The Domain Owner controls this process.

Domains that do not belong to the same home network need to establish trust with each other, i.e., exchange their locally valid Domain CA certificate in order to allow the other domain to authenticate entities registered to the own Domain. This can be done either personally ② or via the Internet ③ using our Trust Exchange services (see Chapter 6). Again, the process is controlled by the Domain Owner.

The Domain Owner can assign access rights to individual entities or to entire Partner Domains ④ using the Guided Policy Administration Point (see Chapter 7). Another component of our system will translate these settings into a valid XACML Policy Set.

Own devices or devices registered to a Partner Domain may now request access to services ⑤. After the authentication of the device using the certificate deployed during registration it will be authorized ⑥ by the local XACML PDP, which uses the XACML Policy Set just described.

The state of the Domain (Domain CA key, settings, exchanged certificates, etc.) is backed up using our Resilience Service ⑦ (see Chapter 11).

Research Questions	State of the Art/ Related Work	This thesis
Guided Security Management System		
Q1/Q2: How to design a user-centric identification/authentication scheme for unmanaged networks based on strong cryptography and how to empower inexperienced users to effortlessly and securely use this scheme?	Public X.509 PKI, GPG/ PGP Web of Trust: Trust and privacy issues, difficult to use. Not suitable for home networking.	Identification/Authentication Scheme (C4.1 - C4.2), Hybrid Trust Model (C4.3), Registration Service (C5.1 - C5.2), Trust Exchange Services (C6.1 - C6.3): User-centered, no trust and privacy issues, easy to use. Adapted to the home.
Q3: How to empower inexperienced users to use a sophisticated authorization system?	Kerberos, Radius, etc.: Centralized, coarse authorization granularity, difficult to configure. Related Works in home network security: Do often not consider user-friendliness, network structure (centralized management) or service access across home networks. Proprietary technology only applicable to the home.	TLS Handshake Interception (C.7.1), Guided Policy Administration Point (C.7.2 - C.7.3): Fine authorization granularity, user-friendly and user-centered creation of access control settings, service access across home networks. Can be integrated in legacy services due to use of standardized technologies (TLS, XACML).
Domain Server and Auxiliary Services		
Q4: How to design a device able to host security services in a home environment securely and user-friendly?	-	Domain Server Architecture (C10.1), Automated Domain Management (C11.1) and Resilience Services (C11.2): Customized device that hosts and protects security services of Domains, user friendly.
Q5: How to protect and guarantee the integrity of this device?	Remote Attestation to and Integrity verification by central server, Secure Boot: Not applicable to the intended scenario, issues concerning trustworthiness, privacy and verification completeness.	Integrity extension of Domain Server Manager, Integrity Verifier Applet on Java smart card (C12.1 - C12.3), Platform Verification Certificate (C12.4): Trustworthy, complete verification, no privacy issues. Customized for the home but generally applicable.
Q6: How to protect keying material used by the identification/authentication scheme against theft and abuse?	Software-based key management and protection: prone to key theft and abuse.	TPM integration concept, TPM manager (C13.1 - C13.2): Hardware-based key protection for Domain CA and entities.

Table 14.1: Overview of findings and contributions and comparison to related work.

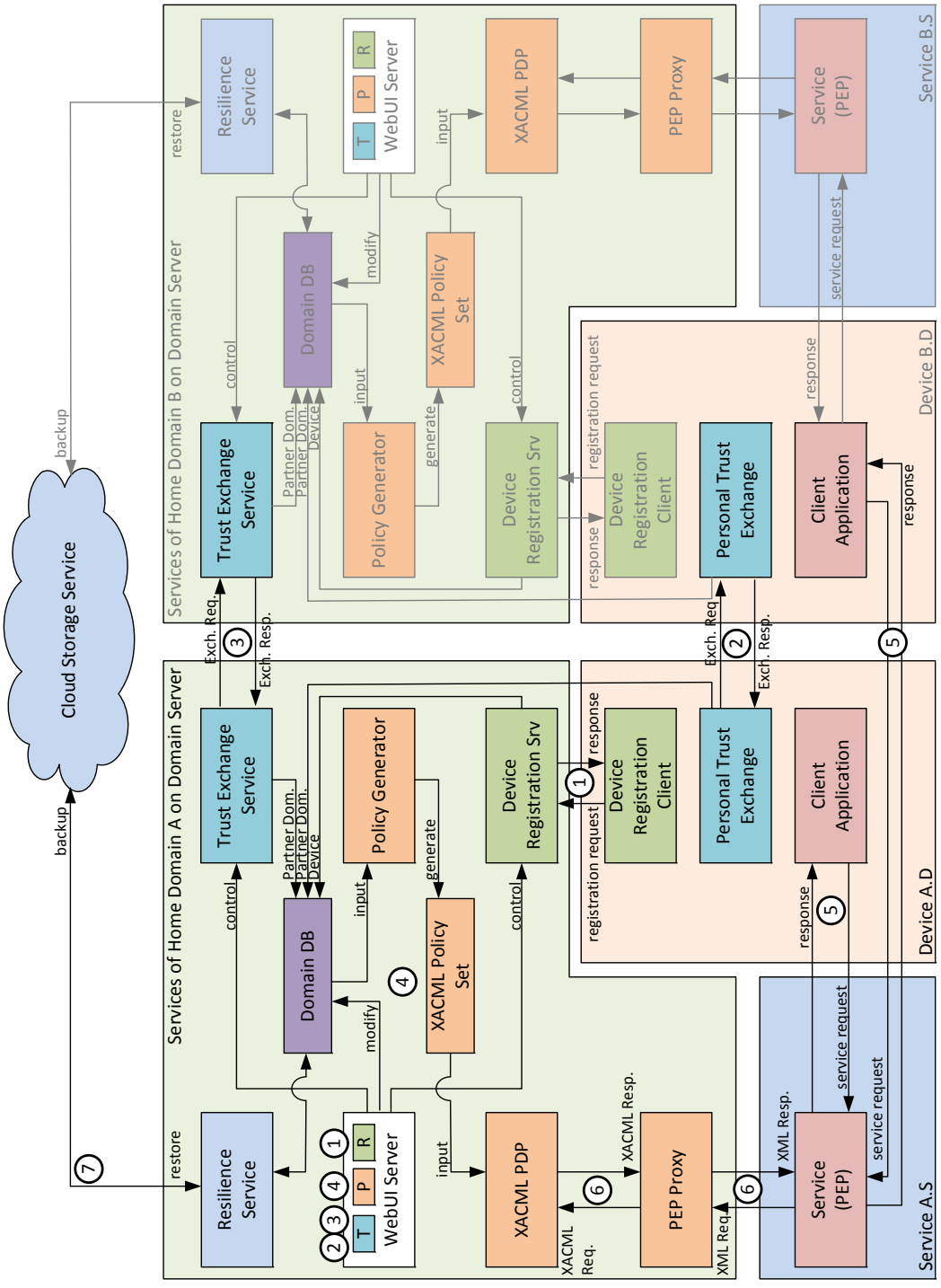


Figure 14.1: Overview on the Access Control System.

14.3 Future Work

We structure the overview on future work by means of the two families of contributions, namely the Guided Security Management System and the Domain Server:

Guided Security Management System

Up to now the potential of TPM-based key protection of Domain CAs or entities is not yet fully exploited. Especially the provability of TPM key protection delivers knowledge beneficial for two interesting extensions of our work:

1) The knowledge about a Domain CA's TPM protection can be used when the Internet Trust Exchange is performed. We argue that CAs with TPM-protected keys are less likely to be impersonated by an attacker than CAs implemented entirely in software. For this reason Counselor Domain CAs with TPM-protection should be preferred over unprotected Counselors Domain CAs. For this purpose, the Trust Metric we created could be made TPM-aware in order to additionally reflect TPM-protected keys. 2) The knowledge about a Domain CA's or an entity's TPM-protection can also be exploited in the context of rights management. TPM-protected Domains might obtain more access rights than other Domains. Individual TPM-protected entities can obtain access rights to especially critical resources. The Guided Policy Administration Point can be extended with functionalities that support this idea.

The Guided Policy Administration Point can also be extended with further functionalities that, for instance, support more complex and fine-grained policies.

We plan to integrate and extend the access control technologies in the Chair's upcoming projects *IDEM (Individualisierbares Energiecontrollingsystem mit dynamischer Mandantenfähigkeit)* and *BaaS (Building as a Service)*. IDEM and BaaS are projects funded by the German Federal Ministry of Education and Research (BMBF)[1] and target smart building environments. User-friendly management of access control settings to IDEM and BaaS services have been defined as one of the goals of these projects.

Domain Server

To guarantee integrity and trust of a device as complex as a Domain Server is a highly complex venture. The mechanisms we created and combined with each other are only able to provide integrity verification at boot time. This means, they measure and verify the integrity of the Domain Server and virtual machines when they are started. This is already highly beneficial, but these mechanisms are not able to deal with attacks that occur at run time, after the device or VM is started.

We discussed earlier that the security of our system can be increased at run time by Host Intrusion Detection Systems and related technologies. These technologies have a weak spot, namely they run in the same environment that they should protect. For this reason they can be attacked and rendered useless by malware. In the Chair's BMBF-funded project *ANSII* [2] *(Anomalieerkennung und eingebettete Sicherheit in industriellen Informationssystemen)* we investigate how run time protection can be integrated into devices with a similar architecture as the Domain Server. One approach we are working on is known as *virtual machine introspection* [94]. The central idea of this technology is to contain protective mechanisms outside the reach of attacking software, i.e., in a virtual machine, and inspect other virtual machines via the hypervisor.

[1] http://www.bmbf.de
[2] www.ansii-projekt.de

This is a highly beneficial technology for two reasons: 1) software attacking a virtual machine is unable to attack the protective mechanisms and 2) protective mechanisms can access the address space of kernel and applications running in other virtual machines from a central place. This finally enables mechanisms that securely monitor a virtual machine at run time and even may interfere when problems are detected. First experiments and results of this research are documented in the Diploma Thesis of Simon Stauber [95].

Part V

Appendix

A. Glossary

This glossary contains short definitions of terms used or of technologies developed in this Thesis. The terms are presented in the approximate order of their appearance in the text.

Identification/Authentication and Registration (Chapters 4 and 5)

- **Public/Private Entity:** An entity is a device or a service. In the home scenario *public* entities are used by all residents, whereas a *private* entity is primarily used by its owner and selected other users.
- **User:** A user is a human being, in the home scenario either the home owner or another resident.
- **Domain:** A Domain is a logical construct that contains all entities owned and managed by a Domain Owner. In the home scenario two Domain types exist: A *Home* (Network) Domain containing public entities, whereas *User* Domains contain private entities.
- **Domain Owner:** The user who owns and manages a Domain. In the home scenario the Home Domain is managed by the home owner, a User Domain is managed by a resident.
- **Domain CA:** A Domain CA is a X.509 Certificate Authority assigned to a Domain. In the home scenario one Home (Domain) CA and several User (Domain) CAs exist. The Home CA has a self-signed certificate and certifies public entities that belong to the Home Domain and User CAs. A User CA is certified by a Home CA and certifies private entities that belong to a User Domain.
- **Cryptographic Identity (cID):** A Domain and all entities registered to it have a cryptographic identity. The cID is derived from the Domain's or entities complete certificate chain. cID are hierarchic, self-certifying and usually have the format of `HomeDomain.UserDomain.Entity`.
- **Human Readable Identity:** A name, email address, etc. that represents a Domain/Domain Owner or entity in a format easily to comprehend by users.
- **Registration:** An entity becomes a valid member of a Domain by registering it to a Domain. From a technical viewpoint, the entity obtains a certified asymmetric key pair for identification/authentication purposes during registration.
- **Registration Service:** The service that assists a user with entity registration.

Definitions of Terms in the Context of Trust Establishment (Chapter 6)

- **Trust** (in a key): The confidence that a certain public key represents a Domain CA resp. Domain Owner.
- **Partner Domain:** A Domain that has performed or wants to perform a Trust Exchange with another Domain.
- **Friendly Domain:** Domains that belong to the local network and Partner Domains are "friendly" Domains.
- **Trust Exchange:** The exchange of Domain CA certificates between two Partner Domains.
- **Trust Exchange Service:** The service that assists owners of Partner Domains during Trust Exchange.
- **Trust Relationship:** A Domain that *trusts* certificates issued by the Domain CA of a friendly CA has a Trust Relationship with this Domain. Domains that belong to the same local network have an implicit Trust Relationship. Trust Relationships between other Domains need to be established explicitly.
- **Identification Level (IL):** A numeric value that measures the confidence a Domain A can have that a certain Domain CA certificate belongs to another Domain B.
- **Reputation Level (RL):** A numeric value that expresses the expectation of a Domain Owner A how much care a Domain Owner B will take when she is performing a Trust Exchange with Domain Owner C.
- **Personal Trust Exchange:** A service that exchanges certificates between Partner Domains during a personal meeting of Domain Owners or their representatives.
- **Internet Trust Exchange:** A service that exchanges certificates between Partner Domains over the Internet with the help of other mutually trusted Counselor Domains.
- **Requester:** The Domain that initiates the Internet Trust Exchange over multiple Counselors with an Invitee Domain.
- **Invitee:** The Domain that should join a Trust Relationship with the Requester Domain.
- **Counselor:** The Domain that acts as Trusted Third Party during the Internet Trust Exchange between a Requester and an Invitee Domain.

Access Control (Chapter 7)

- **Policy Decision Point (PDP):** An authorization server that evaluates authorization requests from a service (PEP) using a XACML Policy Set.
- **Policy Enforcement Point (PEP):** A service offering resources (data, functions, etc.). The PEP uses the XACML PDP to determine if a client is authorized to access the resource and finally enforces the PDP's decision.
- **TLS Handshake Interception:** A technology developed in this thesis that extends a standard TLS Handshake with the ability to authorize an accessing client using a central XACML PDP.

- **PEP Proxy:** An entity that acts as intermediate between a service (XACML PEP) and the XACML PDP of a Domain.

- **Guided Policy Administration Point:** A software component that assists the Domain Owner to implement her security settings as an XACML Policy Set.

Domain Server Architecture and Services (Chapters 10 and 11)

- **Domain Server:** An especially adapted computing device that hosts Security and Service VMs of all Domains that belong to the local network and enforces the network topology needed by the Guided Security Management System.

- **Security Services** are services related to the access control system of a Domain, i.e. the components of the Guided Security Management System and the XACML PDP.

- **Security Service VM:** Security Services are hosted in a Security Service VM.

- **Auxiliary Services** are special services provided by the Domain Server that assist a Domain Owner with setting up her Domain resp. that take care for the resilience of a Domain and the security/integrity of Domain Server and virtual machines.

- **Additional Services** are services not related to the security of a Domain, i.e. services such as network attached storage (NAS), audio/video streaming, home control, etc.

- **Service VM:** Additional Services are hosted in a Service VM.

- **Domain State:** All information that belongs to a Domain is understood as the Domain's state. Examples include keying material, received Domain CA certificates from Partner Domains, information about these Domains and access control settings.

- **Resilience Service:** A service that continuously back ups Domain State to a networked storage service. This service is located in a Security Service VM and initializes the VM with Domain state after finishing its boot process.

- **Domain Server Manager:** A service located in the dom0 of the Domain Server that starts/stops virtual machines when a Domain Owner plugs/unplugs her smart card into/from the Domain Server. This service is extended in Chapter 12 with the ability to verify authenticity and integrity of a virtual appliance resp. a persistently stored VM image.

- **Virtual Appliance:** A ready to use virtual machine image with operation system and software. A virtual appliance is not initialized with Domain state or other personal information of a user.

- **Disposable VM:** A Security Service VM started from a virtual appliance and initialized by the Resilience Service is disposable.

- **Persistent VM:** A Service VM customized by a user is not disposable as customization and private data must be stored persistently.

Integrity of the Domain Server (Chapter 12)

- **Integrity:** *"The integrity of a program is a binary property that indicates whether the program and/or its environment have been modified in an unauthorized manner. Such an unauthorized modification may result in incorrect or malicious behavior by the program, such that it would be unwise ... to rely on it."* [67]
- **Trust** (in a computer system): *"Trust is the expectation that a device will behave in a particular manner for a specific purpose."* [68]
- **Integrity Verifier Applet:** An application located on a Java smart card that verifies the integrity of the Domain Server based on integrity measurement values measured and reported with the help of a TPM.
- **Platform Verification Certificate:** A certificate issued by the Integrity Verifier Applet that asserts that a certain device has passed integrity checks.

Security and Trust for Private Keys (Chapter 13)

- **Identity Key (IK):** The asymmetric key pair that represents the identity of a Domain CA, a device or service.
- **Signing Key:** A TPM key type allowed to sign arbitrary data, such as protocol messages, certificate signing requests, etc. We use Signing Keys as Link Keys and Identity Key.
- **Link Key:** A Link Key is a Signing Key that "links" the non-migratable certificate chain of the TPM to the migratable Legacy Key.
- **Legacy Key:** A key generated in software that is now imported into a TPM. Used as the Identity Key.
- **TPM Manager:** The software component that encapsulates functionality for the Registration Service in order to use the TPM of a device.

B. List of Publications

Subsequently publications of the Author related to this dissertation are listed in the reverse chronological order of the date of their appearance.

- Marc-Oliver Pahl, Heiko Niedermayer, **Holger Kinkelin**, and Georg Carle. Enabling Sustainable Smart Neighborhoods. In 3rd IFIP Conference on Sustainable Internet and ICT for Sustainability 2013 (SustainIT 2013), Palermo, Italy, October 2013.
- **Holger Kinkelin**, Ralph Holz, Heiko Niedermayer, Simon Mittelberger, and Georg Carle. On Using TPM for Secure Identities in Future Home Networks. In Future Internet, 3(1):1-13, 2011.
- **Holger Kinkelin**, Ralph Holz, Heiko Niedermayer, and Georg Carle. On Using TPM for Secure Identities in Future Networks (ext. abstract). In Security in NGNs and the Future Internet Workshop, September 2010.
- Andreas Müller, **Holger Kinkelin**, Sunil Kumar Ghai, and Georg Carle. A Secure Service Infrastructure for Interconnecting Future Home Networks based on DPWS and XACML. In HomeNets: ACM SIGCOMM Workshop on Home Networks, New Delhi, India, September 2010.
- **Holger Kinkelin**, Andreas Müller, and Georg Carle. Security and Access Control for Future Home Networks. Accepted Demo at IPTComm 2010, Munich, Germany, August 2010.
- **Holger Kinkelin**, Heiko Niedermayer, Ralph Holz, and Georg Carle. TPM-based Access Control for the Future Internet (ext. abstract). In 5th GI/ITG KuVS Workshop on Future Internet, Stuttgart, Germany, June 2010.
- Andreas Müller, **Holger Kinkelin**, Sunil Kumar Ghai, and Georg Carle. An Assisted Device Registration and Service Access System for Future Home Networks. In IFIP Wireless Days 2009, Paris, France, December 2009.
- Andreas Klenk, **Holger Kinkelin**, Christoph Eunicke, and Georg Carle. Preventing Identity Theft with Electronic Identity Cards and the Trusted Platform Module. In EUROSEC '09: Proceedings of the Second European Workshop on System Security, pages 44-51, New York, NY, USA, 2009. ACM.

Literature

[1] Jeonghwa Yang and W. Keith Edwards. A Study on Network Management Tools of Householders. In *Proceedings of the 2010 ACM SIGCOMM workshop on Home networks*, HomeNets '10, pages 1–6, New York, NY, USA, 2010. ACM.

[2] Rebecca E. Grinter, W. Keith Edwards, Mark W. Newman, and Nicolas Ducheneaut. The Work to Make a Home Network Work. In *Proceedings of the ninth European Conference on Computer Supported Cooperative Work*, ECSCW'05, pages 469–488, New York, NY, USA, 2005. Springer-Verlag New York, Inc.

[3] A. J. Bernheim Brush and Kori M. Inkpen. Yours, mine and ours? Sharing and use of Technology in Domestic Environments. In *Proceedings of the 9th international conference on Ubiquitous computing*, UbiComp '07, pages 109–126, Berlin, Heidelberg, 2007. Springer-Verlag.

[4] Katrin Busemann and Christoph Gscheidle. Web 2.0: Habitualisierung der Social Communitys. media perspektiven, Heft 7.8.2012, July 2012. Available online at http://www.ard-zdf-onlinestudie.de/fileadmin/Online12/0708-2012_Busemann_Gscheidle.pdf; last accessed on 2013/07/09.

[5] Bettina Klumpe. Geräteausstattung der Onlinenutzer. media perspektiven, Heft 7.8.2012, July 2012. Available online at http://www.media-perspektiven.de/uploads/tx_mppublications/0708-2012_Klumpe.pdf; last accessed on 2013/07/09.

[6] Colin Dixon, Ratul Mahajan, Sharad Agarwal, A. J. Brush, Bongshin Lee, Stefan Saroiu, and Paramvir Bahl. An Operating System for the Home. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, pages 25–32, Berkeley, CA, USA, 2012. USENIX Association.

[7] Colin Dixon, Ratul Mahajan, Sharad Agarwal, A. J. Brush, Bongshin Lee, Stefan Saroiu, and Victor Bahl. The Home needs an Operating System (and an App Store). In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 18:1–18:6, New York, NY, USA, 2010. ACM.

[8] Michelle L. Mazurek, J. P. Arsenault, Joanna Bresee, Nitin Gupta, Iulia Ion, Christina Johns, Daniel Lee, Yuan Liang, Jenny Olsen, Brandon Salmon, Richard Shay, Kami Vaniea, Lujo Bauer, Lorrie Faith Cranor, Gregory R. Ganger, and Michael K. Reiter. Access Control for Home Data Sharing: Attitudes, Needs and Practices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 645–654, New York, NY, USA, 2010. ACM.

[9] Tiffany Hyun-Jin Kim, Lujo Bauer, James Newsome, Adrian Perrig, and Jesse Walker. Challenges in Access Right Assignment for Secure Home Networks. In *Proceedings of the 5th USENIX conference on Hot topics in security*, HotSec'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[10] Patroklos G. Argyroudis. Securing Communications in the Smart Home. In *Procedings of International Conference on Embedded and Ubiquitous Computing (EUC'04)*, pages 891–902. Springer-Verlag, 2004.

[11] Kari Kostiainen, Olli Rantapuska, Seamus Moloney, Virpi Roto, Ursula Holmstrom, and Kristiina Karvonen. Usable Access Control inside Home Networks. *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 0:1–6, 2007.

[12] Jianwei Zhuge and Richard Yao. Security Mechanisms for Wireless Home Network. In *Proceedings of the Global Telecommunications Conference (GLOBECOM), 2003*, pages 1527–1531, 2003.

[13] Max Ziegler, Wolfgang Müller, Rollie Schaefer, and Chris Loeser. Secure Profile Management in Smart Home Networks. In *Proceedings of Sixteenth International Workshop on Database and Expert Systems Applications*, pages 209–213, 2005.

[14] Cor Verkoelen and Harm Schotanus. Access Control and Improved Availability for the Extended Home Environment. In *Freeband Impulse programme*, 2004.

[15] digitalSTROM.org. digitalSTROM. Website, 2013. Available online at http://www.digitalstrom.org/; last accessed on 2013/07/09.

[16] Bosch Thermotechnik GmbH. EasyControl. Website, 2013. Available online at http://www.buderusde/Online_Anwendungen/Apps/EasyControl-3893338.html; last accessed on 2013/07/09.

[17] Miele & Cie. KG. Miele @ home. Website, 2013. Available online at http://miele.de/de/haushalt/produkte/hausgeraete_vernetzung.htm; last accessed on 2013/07/09.

[18] Somfy Systems, Inc. Home Automation - Solutions for everyday living. Website, 2013. Available online at http://www.somfysystems.com/nam/index.cfm?page=/nam/home/discover/our_products/home_automation&language=en-us; last accessed on 2013/07/09.

[19] The ownCloud open-source project. ownCloud. Website, 2013. Available online at http://owncloud.org/; last accessed on 2013/07/09.

[20] Diaspora, Inc. Diaspora - The Community-run, Distributed Social-network. Website, 2013. Available online at https://joindiaspora.com/; last accessed on 2013/07/09.

[21] Wi-Fi Alliance. Wi-Fi CERTIFIED Wi-Fi Protected Setup. White paper, Wi-Fi Alliance, December 2010. Available online at https://www.wi-fi.org/sites/default/files/downloads-registered/wp_20101216_Wi-Fi_Protected_Setup.pdf; last accessed on 2013/07/09.

[22] Stefan Viehböck. Wi-Fi Protected Setup PIN brute force vulnerability. Website, 2011. Available online at http://sviehb.wordpress.com/2011/12/27/wi-fi-protected-setup-pin-brute-force-vulnerability/; last accessed on 2013/07/09.

[23] Jared Allar and Stefan Viehböck. WiFi Protected Setup (WPS) PIN brute force vulnerability. Vulnerability Note VU#723755, United States Computer Emergency Readiness Team, December 2011. Available online at http://www.kb.cert.org/vuls/id/723755; last accessed on 2013/07/09.

[24] heise online. WLAN-Urteil: BGH verlangt marktübliche Sicherung von WLANs. Website, June 2010. Available online at http://www.heise.de/newsticker/meldung/WLAN-Urteil-BGH-verlangt-marktuebliche-Sicherung-von-WLANs-Update-1014360.html; accessed on 2012/08/12.

[25] Digital Living Network Alliance. DLNA - Connect and Enjoy. Website, 2013. Available online at http://www.dlna.org; last accessed on 2013/07/09.

[26] UPnP Forum. UPnP Specifications. Website, 2009. Available online at http://upnp.org/sdcps-and-certification/standards/; last accessed on 2013/07/09.

[27] R. Shirey. Internet Security Glossary, Version 2. RFC 4949 (Informational), August 2007.

[28] James Michael Stewart, Ed Tittel, and Mike Chapple. *CISSP: Certified Information Systems Security Professional Study Guide*. SYBEX Inc., Alameda, CA, USA, 2008.

[29] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008. Updated by RFC 6818.

[30] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. RFC 4880 (Proposed Standard), November 2007. Updated by RFC 5581.

[31] Phil Zimmermann. Why OpenPGP's PKI is better than an X.509 PKI. Website, 2012. Available online at http://www.openpgp.org/technical/whybetter.shtml; accessed on 2012/12/28.

[32] Tim Moses. eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, February 2005.

[33] Tim Moses. eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard, January 2013.

[34] University of Murcia. UMU-XACML-Editor. Website, May 2005. Available online at http://umu-xacmleditor.sourceforge.net; last accessed on 2013/07/09.

[35] Sun Microsystems, Inc. Sun's XACML Implementation. Website, 2003.

[36] University of Applied Sciences Rapperswil. Holistic Enterprise-Ready Application Security Architecture Framework (Heras-AF). Website, March 2005. Available online at http://www.herasaf.org/; last accessed on 2013/07/09.

[37] Blaž Primz. Authenticating Identity Addressing. Diploma Thesis, University of Ljubljana, Ljubljana, Slovenia, 2010. Advised by Marc-Oliver Pahl, Holger Kinkelin, and Andreas Müller.

[38] Simon Mittelberger. Flexible and Resilient Services for User-Centric Identity Management and Access Control. Master's Thesis, Technische Universität München, Munich, Germany, 2013. Advised by Holger Kinkelin and Heiko Niedermayer.

[39] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201 (Experimental), April 2008. Updated by RFC 6253.

[40] The Avahi Team. Avahi. Website, 2013. Available online at http://avahi.org/; last accessed on 2013/07/09.

[41] A. Brusilovsky, I. Faynberg, Z. Zeltsan, and S. Patel. Password-Authenticated Key (PAK) Diffie-Hellman Exchange. RFC 5683 (Informational), February 2010.

[42] Gürcan Karakoc. Ein Mechanismus für nutzerzentriertes Identitäts- und Vertrauensmanagement basierend auf sozialen Netzwerken. Diploma Thesis, Technische Universität München, Munich, Germany, 2011. Advised by Holger Kinkelin and Andreas Müller.

[43] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[44] P. Zimmermann, A. Johnston, and J. Callas. ZRTP: Media Path Key Agreement for Unicast Secure RTP. RFC 6189 (Informational), April 2011.

[45] Paulo Henrique Azevêdo Filho. Guided Policy Administration for Unmanaged Domains. Final Paper of Guided Research Project, Technische Universität München, Munich, Germany, 2013. Advised by Holger Kinkelin.

[46] Dan Driscoll and Antoine Mensch. Devices Profile for Web Services Version 1.1. OASIS Standard, July 2009.

[47] C. Ellison. SPKI Requirements. RFC 2692 (Experimental), September 1999.

[48] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693 (Experimental), September 1999.

[49] Citrix Systems, Inc. The Xen Project. Website, 2012. Available online at http://www.xen.org/; last accessed on 2013/07/09.

[50] Red Hat, Inc. The Kernel Based Virtual Machine. Website, 2012. Available online at http://www.linux-kvm.org/; last accessed on 2013/07/09.

[51] Joanna Rutkowska and Rafal Wojtczuk. Qubes OS Architecture. White paper, Invisible Things Lab, January 2010.

[52] The LVM Project. LVM2 Resource Page. Website, 2012. Available online at http://sourceware.org/lvm2/; last accessed on 2013/07/09.

[53] JTC 1/SC 17. ISO/IEC FDIS 7816-4 – Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. Last updated in 2005.

[54] The OpenSC Project. OpenSC - tools and libraries for smart cards. Website, 2013. Available online at https://www.opensc-project.org/opensc; last accessed on 2013/07/09.

[55] PKCS #11: Cryptographic Token Interface Standard.

[56] PKCS #15: Cryptographic Token Information Format Standard.

[57] Java Card Platform Specification 2.2.2, 2009.

[58] Sun Microsystems, Inc. The Java Card 3 Platform. White paper, Sun Microsystems, Inc., August 2008.

[59] Java Card Classic Platform Specification 3.0.4, 2011.

[60] Java Card Connected Platform Specification 3.0.1, 2009.

[61] Department of Defense. Trusted Computer System Evaluation Criteria. Department of Defense Standard, December 1985.

[62] The Common Criteria. Website, 2013. Available online at http://www.commoncriteriaportal.org/; accessed on 2013/05/17.

[63] Microsoft, Inc. Microsoft seeks industry-wide collaboration for "palladium" initiative, July 2002. Available online at http://www.microsoft.com/presspass/features/2002/jul02/07-01palladium.mspx; last accessed on 2013/07/09.

[64] Richard Stallman. Can You Trust Your Computer?, 2002. Available online at http://www.gnu.org/philosophy/can-you-trust.en.html; last accessed on 2013/07/09.

[65] David Safford. Clarifying Misinformation on TCPA, October 2002. Available online at http://www.research.ibm.com/gsal/tcpa/tcpa_rebuttal.pdf; last accessed on 2013/07/09.

[66] The Trusted Computing Group. Website, 2013. Available online at http://www.trustedcomputinggroup.org/; accessed on 2013/05/17.

[67] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 16–16, Berkeley, CA, USA, 2004. USENIX Association.

[68] Trusted Computing Group. TCG Specification Architecture Overview. Specification, August 2007.

[69] Trusted Computing Group, Inc. TPM Main Part 1 Design Principles. Specification, March 2011.

[70] Trusted Computing Group, Inc. TPM Main Part 2 TPM Structures. Specification, March 2011.

[71] Trusted Computing Group, Inc. TPM Main Part 3 Commands. Specification, March 2011.

[72] Sirrix security technologies. The TrustedGRUB Project. Website, 2009. Available online at http://projects.sirrix.com/trac/trustedgrub/; last accessed on 2013/07/09.

[73] Thomas Müller. *Trusted Computing Systeme: Konzepte und Anforderungen*. Springer Xpert.press, 2008.

[74] Jouni Malinen. hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. Website, 2012. Available online at http://hostap.epitest.fi/hostapd/; last accessed on 2013/07/09.

[75] Maxim Krasnyansky. Virtual Tunnel. Website, 2012. Available online at http://vtun.sourceforge.net/; last accessed on 2013/07/09.

[76] Rafal Wojtczuk. Subverting the Xen Hypervisor. Presented at the 2008 Black Hat Converence, Las Vegas, NV, August 2008.

[77] Rafal Wojtczuk. Preventing and Detecting Xen Hypervisor Subversions. Presented at the 2008 Black Hat Converence, Las Vegas, NV, August 2008.

[78] Zhi Wang and Xuxian Jiang. HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 380–395, Washington, DC, USA, 2010. IEEE Computer Society.

[79] James P.G. Sterbenz et. al. The ResiliNets Wiki - Definitions. Website, 2012. Available online at https://wiki.ittc.ku.edu/resilinets_wiki/index.php/Definitions#Resilience; last accessed on 2013/07/09.

[80] Brian Warner and Zooko Wilcox-O'Hearn and Rob Kinninmont. Tahoe: A Secure Distributed Filesystem. Website, 2013. Available online at https://tahoe-lafs.org/~warner/pycon-tahoe.html; last accessed on 2013/07/09.

[81] Dropbox, Inc. Dropbox. Website, 2013. Available online at https://www.dropbox.com/; last accessed on 2013/07/09.

[82] SpiderOak, Inc. SpiderOak - Zero-Knowledge Data Backup. Website, 2013. Available online at https://spideroak.com/; last accessed on 2013/07/09.

[83] Martin Erich Jobst. Security and Privacy for Virtual Machine Images in Distributed Environments using Smart Cards. Bachelor's Thesis, Technische Universität München, Munich, Germany, 2012. Advised by Holger Kinkelin and Andreas Müller.

[84] Michael Dorner. Combining Trusted Computing and Smart Cards for Trustworthy VPN Access. Bachelor's Thesis, Technische Universität München, Munich, Germany, 2013. Advised by Holger Kinkelin.

[85] Axelle Apvrille, David Gordon, Serge Hallyn, Makan Pourzandi, and Vincent Roy. DigSig: Runtime Authentication of Binaries at Kernel Level, 2004.

[86] Leendert van Doorn, Gerco Ballintijn, and William A. Arbaugh. Signed Executables for Linux. Technical Report UMD CS-TR-4259, University of Maryland, June 2001.

[87] Unified EFI, Inc. Unified Extensible Firmware Interface Specification, Version 2.3.1. Website, April 2013. Available online at http://www.uefi.org/specs/; last accessed on 2013/07/09.

[88] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996 (Proposed Standard), September 2010. Updated by RFC 5998.

[89] PrivacyCA. Website, 2008. Available online at http://www.privacyca.com/; accessed on 2013/05/17.

[90] Simon Mittelberger. A Certification Service for future Home Networks based on Trusted Computing Technology. Bachelor's Thesis, Technische Universität München, Munich, Germany, 2009. Advised by Holger Kinkelin.

[91] Stefan Berger, Ramón Cáceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, and Leendert van Doorn. vtpm: virtualizing the trusted platform module. In *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA, 2006. USENIX Association.

[92] Trusted Computing Group. Virtualized Trusted Platform Architecture Specification. Specification, September 2011.

[93] Trusted Computing for the Java(tm) Platform. Website, 2013. Available online at http://trustedjava.sourceforge.net/; accessed on 2013/05/17.

[94] Virtual Machine Introspection Tools. Website, 2013. Available online at http://code.google.com/p/vmitools/; accessed on 2013/05/17.

[95] Simon Stauber. Vertrauenswürdiges Machine Monitoring durch Hypervisor-basierte Methoden zum Schutz von Netzwerkinfrastrukturen. Diploma Thesis, Technische Universität München, Munich, Germany, 2012. Advised by Holger Kinkelin and Nadine Herold.

ISBN 3-937201-38-6
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)
DOI: 10.2313/NET-2013-10-1