Usability Challenges of PKI

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Math. Tobias Straub

aus Tübingen

Referenten: Prof. Dr. J. Buchmann (TU Darmstadt)
Prof. Dr. G. Müller (Universität Freiburg)

Tag der Einreichung: 13. Dezember 2005
Tag der mündlichen Prüfung: 24. Januar 2006

Darmstadt, 2005
Hochschulkennziffer: D 17

To Judith

Das schönste Glück des denkenden Menschen ist, das Erforschliche erforscht zu haben und das Unerforschliche zu verehren.

– Johann Wolfgang von Goethe

Acknowledgements

First and foremost, I am greatly indebted to Prof. Dr. Johannes Buchmann for proposing the research topic of this dissertation, offering me a scholarship of the Graduiertenkolleg e-Commerce, and promoting and encouraging me while I was working on this thesis. I would also like to thank Prof. Dr. Günter Müller for accepting the task of the second referee.

I had the opportunity to work in a very friendly and inspiring atmosphere at TU Darmstadt. I am grateful to all my colleagues, especially my roommate Vangelis Karatsiolis for everlasting discussions and to Marita Skrobic and Ralf-Philipp Weinmann for their kindness and support. It was a pleasure for me to co-author papers with Harald Baier, Manuel Hartl, Andreas Heinemann, Johannes Ranke, Kai Richter, Volker Roth, and Markus Ruppert. Thanks a lot to the people who proof-read drafts of this thesis: Manuel Hartl, Andreas Heinemann, Carola Klink, and Ulrich Vollmer.

I would also like to thank my family for making possible my academic education and permanently sustaining me. Finally, I thank Judith for her love and patience during the long course of this project. This dissertation is dedicated to her.

Zusammenfassung

Benutzbarkeit ist kritisch für das Funktionieren von Sicherheitsmechanismen. Dies gilt im Besonderen für Verfahren basierend auf Public-Key-Infrastrukturen (PKI). Gerade letztere besitzen eine hohe Komplexität und erwiesen sich daher in der Praxis als Quelle von Bedienfehlern, die schnell zu Sicherheitslücken führen können. Umgekehrt bremsen aber auch Benutzbarkeitsprobleme die Nutzung und Verbreitung von PKI-fähigen Anwendungen und Diensten.

Gegenstand der vorliegenden Arbeit ist die Darstellung der spezifischen Herausforderungen von PKI-Technologie in Bezug auf Benutzbarkeit sowie das Aufzeigen von Lösungsansätzen. Dazu wird ein allgemeines Schichtenmodell zur Verbesserung der Benutzbarkeit von Sicherheitsanwendungen sowie ein generisches Werkzeug zur Analyse beliebiger PKI-fähiger Software eingeführt. Die Arbeit zeigt sodann beispielhaft mehrere Anwendungen dieser Konzepte auf:

Neben technischen Maßnahmen der IT-Sicherheit kommt gerade der Sensibilisierung und Schulung von Anwendern eine große Bedeutung zu, denn viele Sicherheitsprobleme lassen sich auf menschliche Irrtümer oder Nachlässigkeit zurückführen. Wir entwickeln daher ein neues Konzept für Security Awareness-Kampagnen, die durch gezieltes Einbeziehen der Benutzer langfristige und nachhaltige Verhaltensänderungen bewirken sollen.

Das Delegieren von sicherheitskritischen Aufgaben ist eine Möglichkeit, Benutzbarkeitsproblemen zu begegnen. Hier wird speziell PKI Outsourcing, d.h. der Betrieb eines Trustcenters durch einen Dienstleister, untersucht. In diesem Fall ist besonders die Sicherheit beim Certificate Enrolment zu gewährleisten. Wir zeigen eine Sicherheitslücke des Standardprotokolls auf und beheben diese mit Hilfe von verteilten digitalen Signaturen, die ein Vieraugenprinzip kryptografisch umsetzen. Unser Protokoll benötigt eine verteilte Schlüsselerzeugung zwischen zwei Parteien. Wir stellen dazu einen neuen, effizienten und beweisbar sicheren Algorithmus für RSA vor.

Die meisten Emailprogramme unterstützen bereits Verschlüsselung und digitale Signaturen, doch diese Funktionen werden kaum genutzt. Grund dafür ist die ungünstige Kosten-Nutzen-Relation für den Anwender. Wir zeigen, wie sich diese verbessern lässt. Dies geschieht durch den Einsatz opportunistischer Sicherheit, welche Public-Key-Kryptografie ohne Zertifikate und den damit verbundenen Aufwand realisiert.

Ein Problem von PKI-basierten Authentifikationsverfahren ist, dass aus Sicherheitsgründen kryptografische Schlüssel nicht temporär an einen Stellvertreter, etwa während eines Urlaubs, weitergegeben werden können. Wir beschreiben eine, für den Benutzer transparente, einheitliche Lösungsplattform für den Anwendungsfall World Wide Web, die alle gebräuchlichen Authentifikationsverfahren unterstützt.

Wissenschaftlicher Werdegang des Verfassers[1]

10/1995 – 9/1997, 4/1998 – 7/2001	Studium der Mathematik mit Nebenfach Informatik an der Universität Tübingen
10/1997 – 3/1998	Gaststudium an der ETH Zürich
18.07.2001	Abschluss der Diplom-Prüfung (Dipl.-Math.)
8/2001 – 3/2002	Wiss. Mitarbeiter am Lehrstuhl Prof. C. Hering, Fachbereich Mathematik, Universität Tübingen
5/2002 – 4/2005	Promotionsstipendiat im Graduiertenkolleg e-Commerce, Technische Universität Darmstadt
seit 9/2005	Wiss. Mitarbeiter, Fraunhofer-Institut für Sichere Informationstechnologie, Darmstadt

Erklärung[2]

Hiermit erkläre ich, dass ich die vorliegende Arbeit – mit Ausnahme der in ihr ausdrücklich genannten Hilfen – selbständig verfasst habe.

[1] gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt
[2] gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Overview

First things first, but not necessarily in that order.

– Dr. Who

This section starts with a short preface to motivate the subject and to state the research problems and the research agenda. Our contributions are summarized in Section 1.2, an outline of this thesis is given in Section 1.3.

1.1 Preface

The term cryptography, as we use it today, stems from the Greek word $\kappa\rho\upsilon\pi\tau\sigma\gamma\rho\alpha\varphi\iota\alpha$ and its literal meaning of "secret writing". Originally being a technology for governmental or military purposes, the spreading of computers brought cryptography to the desktops of enterprise or home users. The invention of public key cryptography and the idea of digital certificates in the mid-1970's paved the road for the development of cryptographic techniques going beyond encryption. The early concepts emerged to public key infrastructure technology (PKI) which provides the basis for a multitude of security services and applications. In the early 1990's public key cryptography hit the mass market with the software package PGP (Pretty Good Privacy), which allowed end users to sign and encrypt their email and data with the strongest algorithms. Today a lot of commercial off-the-shelf applications – like email clients, web browsers, file encryption software, groupware, VPN clients – have some form of built-in PKI support. PKI has also quietly seeped into consumer devices, which are now equipped with powerful

processors and network capabilities, for example fourth generation mobile phones.

Internet users regularly get into contact with public key cryptography in the form of SSL (Secure Socket Layer), which provides server authentication and secure communication channels. Most of them are not aware of this until warning or error messages pop up. However, a significant part of users does not react correctly in such a situation or is able to tell apart secure connections from insecure ones [88, 238, 119]. Usability issues are neither a new, nor a specific problem of cryptography. Consider the use of cryptography during the Second World War as an example: It is known that Russian soldiers abandoned the official army ciphers because they were too difficult to use and switched to a simpler system, which proved easy to break [135]. On the other hand, users perceive computer security in general as a barrier that gets into their way when pursuing primary tasks (see e.g. [4]). Consequently, they pay little attention to security measures or even deliberately try to short-cut them. In fact, a majority of today's security breaches can be traced back to human error [206, 245, 142].

Interestingly, the demand for usability in security applications dates back quite a long time. Usability was an explicit requirement stated by Saltzer and Schroeder [199] as one of their eight design principles for secure systems thirty years ago. But people still have difficulties in applying security mechanisms correctly as numerous user studies show [253, 4, 204, 88, 98, 251]. A significant amount of work on usable security was done in the last decade. For instance, problems related to password usage were studied intensively as passwords are the most common authentication method (see e.g. [202] for a survey). Research on usable security also covers topics ranging from group collaboration and file sharing [71, 18, 110], identity and privacy management [133, 132, 38] to email security with PGP [253] or S/MIME [94, 92], and the protection of wireless networks [17]. All these application are related to PKI since they use cryptography to achieve their security goals. However, previous research has only concentrated on studying partial problems.

Problem Statement There has been no systematic approach yet to study usability issues of PKI-enabled applications explicitly. By PKI-enabled we mean software that implements security mechanisms with public key methods. We believe that a dedicated analysis is worthwhile for three reasons.

- In the first place, PKI-enabled applications have the same security primitives and technical mechanisms in common. They all have to deal with recurrent tasks like key and certificate management or trust decisions and especially the question of how to design the corresponding user interfaces and interaction mechanisms.
- PKI is inherently complex – too complex in fact for users of current PKI-enabled software, who have very little technical background [88, 60]. Left alone with the usability challenges of PKI, they may quickly open security breaches [253, 238]. This is critical against the background that the number of users, e.g. of digital signature SmartCards, will grow in the near future [144].
- Finally, public key cryptography itself offers a wealth of technologies that can help to address usability problems. Up to now, these technologies have mostly been looked at from a security perspective. Cryptography is a young science, so common practices for application design should be scrutinized and not be regarded as given.

Research Agenda Our analysis implied the following research agenda:

- Describe what usability problems in secure software in general are and where they originate from,
- point out the particularities of PKI-enabled applications,
- analyse generic strategies to improve the usability of secure software,
- and show by means of examples how these techniques can be successfully applied to PKI-enabled applications.

1.2 Contributions

This dissertation is based on the thesis that the usability of PKI-enabled applications has to be looked at from different angles. On the one hand, PKI is a security technology and as such inherits usability challenges that are characteristic for secure applications. On the other hand, PKI brings in additional complexity. We show how to cope with these challenges. The contributions of our research include the following:

Concepts and Techniques

- A multi-layer model of methods to promote usable security and a classification of previous work according to this scheme.
- A generic and flexible tool to evaluate the usability and usefulness of PKI-enabled applications of any kind.
- A new approach for information security awareness campaigns called *Awareness by Doing*.

Experimental Results and Artefacts

- A tool for the quantitative analysis of users' mailboxes suggesting the feasibility of opportunistic security with manual key verification.
- A paper-based user test showing the understandability of a new metaphor for secure email.
- *TLS Authentication Proxy*, a platform to handle delegation of WWW credentials seamlessly and securely.

Cryptographic Results

- A modular protocol for certificate enrolment in an (outsourced) PKI enforcing a four-eyes principle with cryptography.
- A distributed RSA key generation algorithm for the two-party case with only linear asymptotic complexity.

Parts of this dissertation were developed in parallel and published independently. References to the corresponding publications of the author are given below.

1.3 Outline of this Thesis

This dissertation is divided into two parts. The first one concerns itself with usability challenges for security software in general and PKI-enabled applications in particular. In the second part a number of technical solutions are presented that intend to simplify the users' task. A detailed overview of the remaining chapters is given in the following.

Part I

Chapter 2 provides examples and an introduction into the topic and identifies reasons for usability-related weaknesses of security software. We present a multi-layer model of methods to improve usability. This model guides us throughout the whole thesis as our solutions are examples for one or more of those layers (see Figure 1.1). We review previous work and classify it according to our scheme.

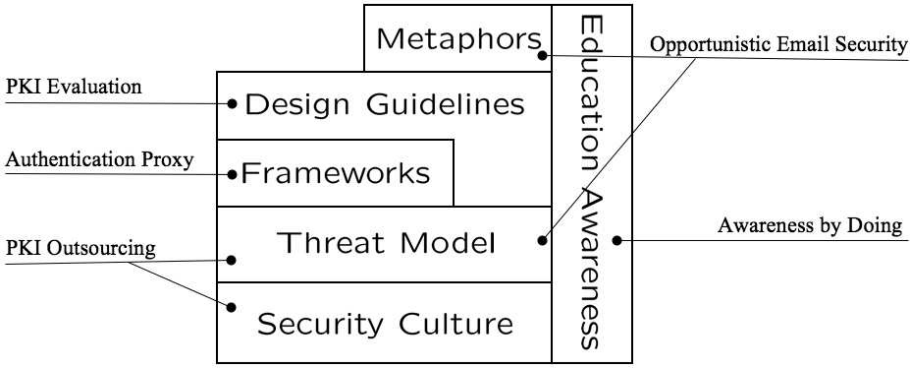

Figure 1.1: Thesis outline.

At the beginning of the third chapter we explain the main concepts of PKI for non-specialist readers. We identify and explain a number of challenges that may make PKI difficult to use. The essential part of Chapter 3 is our evaluation framework for measuring the usability and usefulness of PKI-enabled applications. Parts of this work were published in [233]. The framework was applied to a variety of end user applications and trustcenter software [44, 151].

Part II

A couple of solutions to the manifold usability challenges are the subject of the second half of this dissertation. User education and awareness training is an important axis to lower security-critical user errors. In Chapter 4 we propose a new method for information security awareness training towards more efficient and sustained changes in users' minds and actions. *Awareness by Doing* was first presented in [16].

The delegation of critical tasks to skilled personnel or a service provider may form a way out of usability problems as well. But this must not happen at the price of lower security, especially when talking about PKI outsourcing. For this purpose, we propose a new certificate enrolment scheme [227] in Chapter 5 that avoids a critical weakness in the standard scheme. The cryptographic ingredients to our recipe are shared digital signatures and distributed key generation. We present a new algorithm for distributed RSA key generation [226] which is more efficient than previous ones. Finally, we show how our software library for threshold cryptography can be applied to the use case of secure email [234].

Email is a key application on the net today. Although most clients are PKI-enabled, only a vanishing portion of current email is sent encrypted and/or digitally signed. This is due to the unfavourable cost-benefit ratio for users. Making key verification optional we apply the idea of opportunistic security to improve this ratio. Empirical data we gathered during the work on this subject [193] provides strong evidence that this should indeed be feasible for most users. A generalisation of the opportunistic security concept is conceivable, e.g. to SSL [228].

An increasing number of web sites uses certificate-based client authentication. An important drawback of such credentials is the fact that they cannot be easily delegated without an appropriate infrastructure. Practical needs, however, demand that people can temporarily share credentials, e.g. when going on holiday. In Chapter 7 we present *TLS Authentication Proxy* as a unified platform to support the delegation of credentials. It supports the common authentication methods found on the Internet today. TLS Authentication Proxy has been the subject of two publications [104, 235].

Finally, Chapter 8 summarizes the conclusions of the thesis and provides an outlook towards further directions of research.

Part I

Analysis

Chapter 2

Usability vs. Security?

> *Secure systems have a particular rich tradition of indifference to the user, whether the user is a security administrator, a programmer, or an end-user.*
>
> – Mary E. Zurko

> *Going on 9 full years after I generated my first PGP key, my mom still cannot use the stuff.*
>
> – Adam Shostack

System designers and users often regard security and usability as being two conflicting goals as security software has traditionally been difficult to use [260]. There is a significant difference between civil environments and the military, where cryptography originates from. In the latter case a well-trained personnel is highly motivated (at least by the threat of punishment) to adhere to security policies and to take great pains over the software. As in the early years of computer security, the military use cases were predominant, only little incentives existed for software makers to produce usable applications. Even worse, this led to a very formalized view (e.g. in the Bell-LaPadula model [22]) towards system security omitting the user as a main "component" [260, 39]. Actually security and usability, however, can be seen as aspects of a common goal, namely fulfilling user expectations [259].

From the perspective of a software engineer usability requirements add additional challenges to the development process of security software. From the viewpoint of an end user security measures come into her or his way when trying to accomplish productive tasks. In a broader view, software features

and development costs are another two design goals that are at odds both with security and usability. The situation is depicted in Figure 2.1 where corners of the tetrahedron symbolize conflicting design goals. This drawing

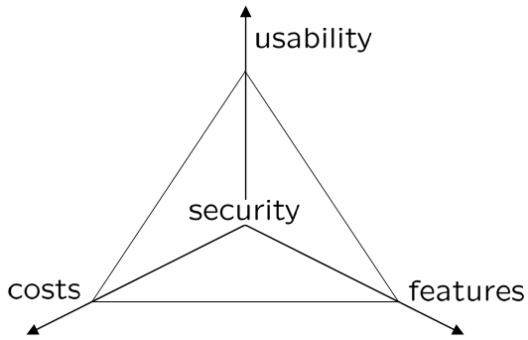

Figure 2.1: Divergent design goals for secure software.

suggests that one cannot "be at all corners at the same time". Of course we mainly confine ourselves to the perceived antagonism between usability and security in the following, so we only say a few words about the other relations. Without doubt, economic aspects are an interesting subject as well, because the lack of proper business cases was and still is a hurdle to PKI adoption, too [7]. However, a thorough treatment is out of the scope of this thesis. Readers interested in this area are referred to [11, 122, 241, 130, 217] as starting points for further reading.

The tension between usability or security on the one hand and features on the other hand is more implicit, but seems easier to tackle. For instance, common HCI (human-computer interface) methods can be applied to allow a system to be rich in features (both "productive" as well as cryptographic ones) and simultaneously usable. Standard techniques like hiding complexity, different user levels, or multiple modes of control can be applied (see e.g. [19] for an overview). Trade-offs between security and features are typically made at the outset of the design process, e.g. by the statement of security policies [70].

In this chapter we first describe the big picture of the communication model and the security goals in an abstract fashion. We assume a basic familiarness of the reader with the topic of PKI and cryptography in this introduction. In case it is necessary, the reader is referred to additional technical information that can be found at the beginning of Chapter 3. A num-

ber of examples of security incidents are included in Section 2.2 to motivate the subject. These examples led us to the typification of usability-related weaknesses of secure applications. The last section extends the presentation of Section 2.2 and sets forth a multi-layer model of methods to improve usability. Related work in the area of usable security software is classified according to this scheme.

2.1 Security in Open Networks

In this section, we introduce the standard network model which is typically used for the description, analysis, and design of secure systems. For the purpose of discussing usability issues, an intuitive, non-formal notion is mostly sufficient. We will take on a more formal and technical view where applicable (Chapters 5 to 7).

The main characteristic of the network model is the property of *openness*, which is a common assumption [8, 70, 219, 215]. Here, openness means that communication relationships are not predetermined from the outset and may change quickly over time. As we have stated before, secure communication is now a key requirement of ordinary people's Internet transactions, mobile phone connections etc. as it has left the military environment. The latter one is the archetype of a "closed network", i.e. it has a restricted user base and/or spatial extension, and possibly its own standards and protocols [70].

2.1.1 Network Model

Throughout this dissertation we consider threats and countermeasures to IT (information technology) systems that consist of computers in an *open network*. When talking about PKI one often has Internet-based, electronic commerce applications in mind where users buy goods from previously unknown web sites or get into contact with people they never meet in person etc. (Adams and Lloyd [8] call this "security between strangers".) Here openness refers to the idea that communication relations are variable and not known a priori. We somehow abstract from this idea and regard an *open network* as a set of computers which may be connected pairwise by zero or more information *channels*. We use the term *message* to denote the data that can be transferred over a channel, for instance emails, instant messages, or files of an arbitrary type.

The notion of "computer" may in practice cover a broad spectrum from a powerful high-end server in a computing Grid, a network router to a PDA (personal digital assistant), a mobile phone or a device like an embedded system with very limited resources [213]. Where a distinction between computers and their owners is not made, the term *network entity* or simply *entity* is used instead. Often we assign names (Alice, Bob, Carol etc.) to these entities. "Channels" may take on the form of wired Internet connections, a wireless local area network (WLAN) connection, a telephone line, a short-range (e.g. Bluetooth, IrDA) or out-of-band (e.g. a human being carrying storage media from one machine to another) connection – to name some examples. Channels can be temporary or permanent in nature, they may correspond to a physical (e.g. cross-link Ethernet cable) or logical (e.g. virtual private network, VPN) link. Messages are transferred along channels and encoded in an appropriate *protocol* which the entities at both ends have to understand and follow.

To have an example, one can think of a number of different types of entities – personal users, a university, an enterprise, an Internet service provider (ISP), an Internet shop, and a trustcenter – and some selected communication channels between them – say Internet, WLAN, LAN (local area network), VPN, Bluetooth, IrDA. We deliberately do not distinguish between physical and logical channels here. When drawing network graphs, they are usually simplified by using a "meta vertex" called "Internet" which hides the fact that there are numerous machines in between two systems connected via the Internet. Other intermediaries are also removed to reduce complexity. Obviously, in order to analyse the security of a particular use case, the respective part has to be refined accordingly. This is done in Chapter 5 and 6 for the internal network of a trustcenter and Internet connections for email respectively.

2.1.2 Security Threats

Security threats and attacks are typically described using the setting of two or more network entities connected by channels [219]. A distinction can be made between threats and attacks in the following sense. A threat is a potential danger whereas an attack is a concrete assault on system security [212]. An attack is mounted by an *attacker*. This notion expresses the difference between safety and security: Safety means protection against

accidental events while security means protection against intentional actions (cf. Section 2.3.1).

Consider two entities, say Alice and Bob, that are connected by a single common channel. There are four threats to this channel and the two parties: An attacker might ...

(1) *eavesdrop* on the channel to learn the messages that are exchanged by Alice and Bob,

(2) *block* the channel such that Alice and Bob cannot exchange messages with each other,

(3) *generate* a message on her own and trick one of the two parties into believing that the message comes from the other party,

(4) *modify* messages from Alice to Bob (or vice versa) before passing them on to the intended entity.

There is another "attack" or form of misconduct, which only involves the two parties themselves. Alice or Bob may

(5) *repudiate* having send (or received) a certain message to the other party.

An *attacker* (or *adversary*) can be understood as a party that temporarily or permanently affects one or more edges of the network graph and takes on the actions above. The attacker is characterized depending on her *power*. She is called *passive* if the only thing she does is eavesdropping and *active* otherwise. An active attacker is also called a *man-in-the-middle* (MITM), because of his power to control the whole communication between Alice and Bob. Attackers are often given meaningful names such as Eve ("eavesdropper"), Trudy ("intruder"), or Mallory ("malicious") to point out their role in the game.

2.1.3 Security Goals

The threats of the previous section can be abstracted from in order to formulate a handful of generic goals for IT security. A historical definition [246] captures the three goals of confidentiality, integrity, and availability. Nowadays integrity is subdivided in authenticity, non-repudiation,

and integrity in a narrower sense (also called data integrity) in the literature [42, 70, 219, 212]. We use the following, standard terminology of protection goals (see Table 2.1). These goals correspond to the threats listed above (in the same order). When speaking of public key cryptography and public key infrastructures, these are the protection goals one usually has in mind. We now give a formal definition of what is a PKI-enabled application (an introduction in public key cryptography is given at the beginning of Chapter 3).

Definition 1 (PKI-enabled application) *An application is called* PKI-enabled *if it implements security mechanisms with methods from the domain of public key cryptography.*

Note that our definition does not mean that the application has to conform to certain standards in this area (like X.509 [52] or PKCS[1]), nor does it necessarily imply that certain security goals are pursued, nor does it prescribe that the application processes certificates at all. In this sense, SSH (Secure Shell) clients are considered PKI-enabled as well as servers that use an anonymous Diffie-Hellman key exchange or SmartCard-based applications for the creation of digital signatures.

Goal	Description
Confidentiality	Protect data from being disclosed or made available to unauthorized entities.
Availability	Ensure that authorized users get systems services they demand for.
Authenticity	Ensure the genuineness of a message (authenticity of data origin) or verify the identity of a peer (entity authentication).
Integrity	Assure that data cannot be modified in any way without being detected.
Non-Repudiation	Prevent that parties can deny having sent (or received) a certain message.

Table 2.1: Common protection goals in information security.

Please note that this list of protection goals is neither complete, nor are all of those goals considered of the same importance. Things strongly

[1] `http://www.rsasecurity.com/rsalabs/node.asp?id=2124`

depend on the concrete use case. For instance in the context of contract signing and the like, non-repudiation is a vital goal, whereas in secure web browsing it is not. Additional or even conflicting protection goals are also involved.

Confidentiality extends to *privacy*, i.e. the need of an entity to act in a way such that she is unobservable by others and controls the amount of information that is shared with others. Data confidentiality usually refers to the protection of payload, while privacy also covers traffic flow information. Non-repudiation is also called *accountability* or *content commitment* (according to X.509 PKIX terminology [128]). *Accountability* in a broader sense may be understood as a system property which allows tracing all actions back to an entity [212]. Obviously, accountability and privacy are at odds with each other. The cryptographic technique of so-called ring signatures [190] and their application, e.g. for secure email may be an interesting area of further research.[2] A ring signature can be used to authenticate a message, but with the property of *repudiability*. That means that the recipient cannot prove the authorship of the sender to a third party, a feature comparable to using a message authentication code (MAC, see page 62) instead of a digital signature (which however requires more than one round of communication).

Often *authorization* is added as another main security goal, although not being a property of the channel. Authorization refers to the demand of ensuring that subjects (entities) only perform actions on certain objects (computing resources, files etc.) at certain circumstances (time, location etc.) they are allowed to. Authorization requires prior subject authentication.

The fact that users do not necessarily know and trust each other, especially in open and complex environments was emphasized in [188]. This approach is called *multilateral security*. A refinement of the security goals listed above is made in [133] for the use case of identity management, i.e. the protection of one's personal data in an online world. A distinction is made with respect to the possible attacker, by separating the protection against all others and the protection against third parties while considering communication partners as trusted. Dependencies and implications among goals like

[2] The author thanks Volker Roth for discussions on this aspect.

unobservability and anonymity (which are both a facet of confidentiality) can be proved [99].

2.1.4 Usable Security: Why and How

In this section and the next one, we explain the role of usability in the context of security-related applications.

As a general taxonomy we use the one due to Nielsen [173]: *Usability* and *utility* are considered common sub-categories of the category *usefulness*. Usefulness and characteristics like costs or reliability are aspects of a system's *practical acceptance*. The question whether a system, in principle, has the functionality necessary for a certain task is a matter of utility, whereas usability refers to how this functionality can be used in practice.

We use the definition of usable security which is due to Gerd tom Markotten [97]. It is derived from general usability guidelines like those given in ISO 9241 [131]. "Usability of security" typically covers "usability of security-related applications", but is not be restricted to that [215]. It is also important to have in mind new approaches for the integration of security into applications as well as new security technologies that enhance usability.

Definition 2 (Usable Security) *Usable security is the degree how efficiently, effectively, and satisfyingly an end user can protect himself and his IT system in a certain context.*

Usability Evaluation A rich arsenal of usability evaluation techniques is known, testing (laboratory testing, discount usability testing) and inspection methods (heuristic evaluation, cognitive walkthrough) being the most prominent (see e.g. [173, 176] for an in-depth treatment). There is a certain controversy about whether security needs its own, different usability standard. Both, established usability methods [260, 204, 136, 127] as well as methods adopted to the security setting [97, 253] have been advocated and successfully applied. Usability in general can be measured e.g. by means of user performance (i.e. referring to the time necessary for a certain task or the error rate) or user satisfaction (figures usually provided by surveys or interviews). However, in a security setting, these metrics have to be taken with a grain of salt since they are tailored for standard end user applications. To overcome this problem, several methods have been proposed. Among

those are data logs that run in the background of the application and monitor which pieces of data were protected by the user and how. Some care has to be taken with user tests and interviews for the usability of security applications (see [215] for details).

Taking the Human Factor Seriously Just as computer security mechanisms are more than pure cryptography, system security is more than pure technology. Security has a number of facets ranging from the lower layers like cryptography and network protocols, security models to implementation aspects and the human-computer interface, security policies and social engineering. A holistic approach is necessary as an attacker may try to compromise any of the components of the security layers. There is the famous proverb "security is like a chain" in that the weakest link determines the overall security of the system.

Strong evidence from the field [206] indicates that the human factor is in fact this weakest link or that at least designers of secure applications should assume this as a working hypothesis. It has been proposed to always assume *a priori* that the user is the weakest link when analysing and designing security mechanisms or tools [100]. Very often system security is undermined because of human error or misconduct as studies show (see e.g. [75, 245, 142]. Some interesting quantitative findings of these studies are cited at the beginning of Chapter 4.

In Section 2.2, we give a handful of examples where secure applications (or security mechanisms of other applications) fail. Examples include passwords and Internet security as well as the handling of secure email or online privacy. We do not aim to present a comprehensive list of current threats here and cover only those which are related to cryptography and PKI. PKI-enabled applications have to cope with common challenges as they are based on the same conceptual and technological foundation. We list the most urgent ones of those challenges in Section 3.2.

Reasons for Usability-Related Security Weaknesses In the following we identify and categorize usability-related weaknesses of applications. A first coarse distinction can be made with respect to the question whether users compromise security intentionally or unintentionally, a canonical criterion [98].

Who is Responsible? We speak of "user-caused violations" and "application-caused vulnerabilities" and distinguish four reasons in each group. The terms "violations" and "vulnerabilities" merely refer to the same thing. Here we use different words to express the user's role as a subject and the application's role as an object. Detailed descriptions can be found in Figures 2.2 and 2.3. User-caused violations can be ordered according to growing intent. Violations which are due to a lack of vigilance are the most unintentional ones whereas actions of a malicious attacker are on the other end of the scale. Very similar, application-caused vulnerabilities are ordered from relative to absolute impossibility. Here we have "sloppiness" on the one extreme and "left as an exercise" on the other.

Security-Inherent Properties Affecting Usability Having categorized a number of reasons for usability-related security problems, we now go one step further and look at inherent properties that make "security" conceptually difficult. Whitten and Tygar [253] name five characteristic properties which are shown and explained in detail in Figure 2.4. These properties have to be addressed both by software engineers and user interface designers. First of all, security is typically a *secondary goal*, so users are unmotivated to pay too much attention to it and tend to use the method of "trial and error" [133, 99]. Besides, cryptography and PKI are *abstract* and complex matters, therefore designing applications that give appropriate *feedback* is challenging. This can also be called the "hidden failure property". Security applications often have the characteristic that mistakes decrease the level of security in a way one cannot completely recover from. Whitten and Tygar use the term *barn door property* for this aspect. As a consequence, irrevocable security mistakes must be ruled out. We have already mentioned the common sense that security is like a chain, in that the weakest link determines the overall quality of the system. Special attention is thus paid to the *weakest link property*.

Blame Designers, Not Users We conclude that users are not the (only) ones to point the finger at in the security game. Security in general and PKI in particular bears a lot of technical challenges. Some of those are in fact hard to solve properly and elegantly. We have referred to these issues in Figure 2.2 and Figure 2.3. One might argue that training and education can help to defuse the risk of user-caused violations. This assumption is

Users *violate an explicit security policy rule* because they ...

unintentional

(slips) are careless or even curious when they are in a situation where they should (and do) know better. Mostly these are familiar and routine tasks, like opening an email attachment or installing software downloaded from the Internet. These actions fall into the category of *slips*, a particular kind of human error. The book of Norman [179] provides a finer classification of slips as well as design guidelines to avoid them.

(perception) underestimate their personal behaviour's impact on the overall security of their PC, business processes, or their organization etc. This originates from the perception that security standards in the organization are low anyway or from underrating security threats [204]. Information security awareness is addressed in more detail in Chapter 4.

(work-around) consider the existing policy obtrusive. Reasons for this are the cost-benefit ratio which is perceived as being poor or the fact that the policy does not match the way they actually work with the system (e.g. co-workers that share a password in order to have access to common project files). In order to get their tasks done, they feel the need to work around security mechanisms (see e.g. [204]).

intentional

(malicious intent) want to weaken the system in order to cause damage. A fired employee taking revenge is a typical example. This category though is not of interest from a usability point of view. We do not consider it any further.

Figure 2.2: User-caused violations.

valid for the professional or organizational context, but does surely not hold for home users. A major obstacle is the fact that security is not a primary goal, particularly not for such users. This suggests that other solutions are necessary. In Section 2.2 we present and discuss several approaches from the current body of research. These approaches are high-level design proposals and as such apply to general security mechanisms. Particular attention to PKI-enabled applications is paid from Chapter 3 onward.

The application makes it *hard to operate it properly* because it ...

relative impossibility

(sloppiness) is implemented in a sloppy way. As a consequence, particular tasks cannot be carried out suitably due to general usability flaws. A poor implementation of the user interface decreases confidence in the underlying security mechanisms, too [98]. Albeit sloppiness is not per se a security-related flaw, it plays an important role in this context as the perceived trustworthiness directly relates to the willingness to use the system [137].

(expertise gap) requires a deeper understanding of the underlying security principles. The intended security goals cannot be accomplished correctly or efficiently in practice because users lack previous knowledge, which the application takes for granted. PGP 5.0 was a negative example in this respect, being symptomatic for PKI-enabled applications [253, 60]. Another issue which a lot products suffer from is the use of "slang", i.e. very technical or even inaccurate terms.

(no fail-safe defaults) permits users to change the configuration without warning them of insecure settings or even comes with a default configuration that is prone to vulnerabilities. This could be clearly seen in the KaZaa application [110]. Fail-safe defaults have already been postulated as an important design guideline long ago [199] and emphasized for usability reasons [100]. However this aspect of error prevention is often neglected.

absolute impossibility

(left as an exercise) leaves the user alone in ticklish situations which can be dangerous. On the one hand, applications often force the user to make an immediate choice or decision, but do not provide appropriate guidance or the alternative to postpone the action. On the other hand, the majority of applications shirks addressing (or even making clear) PKI tasks that are hard or to be solved "out-of-band" (e.g. certificate lookup and path construction or key fingerprint verification).

Figure 2.3: Application-caused vulnerabilities.

(P1) Unmotivated user property Since security is hardly ever a user's primary goal, she cannot be expected to pay too much attention to it or, for instance, go through voluminous manuals before using the software.

(P2) Abstraction property Especially in the field of public-key cryptography and PKI, it is a difficult task to find intelligible yet precise (real world) metaphors for abstract mathematical objects like key pairs or certificates.

(P3) Lack of feedback property Applications should above all enforce security in a seamless way. In cases where interaction is required, the user should be provided with the necessary feedback to allow for an appropriate decision. Presenting a complicated matter in a short dialogue is challenging.

(P4) Barn door property Dangerous and irrevocable mistakes must be ruled out from the start. If a secret has been revealed at a certain point of time, say during an insecure network connection, one cannot say whether an attacker might already know it.

(P5) Weakest link property Security is considered to be a chain the weakest link of which determines the strength of the whole system. As a consequence, users need to systematically and comprehensively take care of an application's security settings.

Figure 2.4: Security-inherent properties (identified by Whitten and Tygar [253]).

2.2 Examples and Previous Work

As not to distract the expert reader from the analysis in the previous and the next section we chose to treat examples in a separate section.

2.2.1 Passwords

A lot of research has been done on the usage of passwords since they are a widespread security mechanism. A significant part of this work [5, 4, 250, 204, 203, 202, 39] was done by Angela Sasse's group at University College London. Empirical data about usability problems with password usage is found, for instance, in [79, 187, 204, 257].

Typically, a password is used for user authentication in combination with a user (alias) name which serves as identifier. User names are typically not kept secret, the security of the mechanism relies solely on the strength of the secret shared between user and the system. Passwords may seem an old-fashioned method in the light of biometrics and PKI, but passwords are to stay here for several reasons. On the one hand, passwords are often sufficient for low-risk environments or low-valued transactions where PKI and biometrics would surely be oversized solutions (cf. the discussion in Section 2.3.2). Biometric authentication is not ubiquitous as special hardware (fingerprint readers, iris scanners etc.) is required, a drawback that also applies to PKI when SmartCards are used.

On the other hand, passwords still play an important role even in a PKI. Passwords are a common mechanism for the protection of certificate stores (integrity) or private keys (confidentiality or authorization, see below). Consider the case where a person wants to use her private key to decrypt a ciphertext or sign a message. If the private key is stored in hardware, e.g. on a SmartCard, the user has to enter a PIN[3] and provide it to the device. The device requires the correct PIN as an authorization. It then carries out the requested operation with the private key and returns the result. In case the private key is stored in a *softtoken* a bitstring is derived from the password. This bitstring serves as a symmetric key to temporarily decrypt the private key in memory (an example for a password-based encryption mechanism is PKCS#5). Another example is the use of pass-

[3] A *personal identification number* which is basically a password with a fixed length (e.g. 4 or 6 characters) and a small, numeric alphabet.

words – or "passphrases" as they are called in that context – in the PGP community. Here the term "phrase" instead of "word" suggests that longer character sequence are (or should be) used as PGP users often emphasize an individual's privacy protection.

Several problems with passwords were reported repeatedly over the past decades [79, 4, 187]. Looking at security considerations first, it can be noticed that a significant portion of passwords used in the field is weak either because words from a dictionary are used or because of the passwords' insufficient complexity (with respect to length, alphabet and/or randomness). Password cracking tools like "John the Ripper" apply a combination of dictionary attack (checking a large set of words found in a dictionary and variations thereof) and a brute-force attack (probing all possible passwords up to a certain length) techniques [16]. The survey [187] revealed that 40% of the user population have shared passwords with co-workers once or multiple times and that the majority of users (55%) tends to write passwords down, a "radical" fraction (9%) even writes down *every* password they have to use. These figures are confirmed by a recent *Meetbiz Research* survey. Half of the users have at least once shared a password with a colleague, 15% even multiple times [2]. Obviously, such tendencies make password policies very vulnerable to "social engineering" attacks. Passwords are also in the focus of the new IT security awareness approach outlined in Chapter 4.

There are numerous reasons why the password security mechanism does not work as intended in practice. A crucial constraint is human memory. The policy that different passwords should be used for different accounts increases the number of passwords in use significantly. People in general have difficulties in exactly recalling a large number of passwords possibly consisting of meaningless strings. For instance, employees were found to use 16 passwords on average in the study conducted by Sasse *et al.* [204]. Difficulties are increased by policies that dictate periodical changes and enforce complexity rules – both may vary from one account to another. As accounts are used infrequently, passwords can only be recalled partly or are confused with a previous password of the same machine or that from another account.

Various "graphical password" schemes have been proposed and evaluated in the last few years (see [65] for a survey). These schemes rely on human's ability to recall series of images which is superior to that of recalling se-

quences of random characters. Plus, some of the schemes are fault-tolerant in that they accept user answers that are sufficiently "close" to the secret.

The story of passwords illustrates an important problem of security mechanisms in general, namely the fact that they do not match the way people actually work with the system. For instance, users that work together on a project typically have to share their data. If their system does not permit to share files easily (or does not tell users how to configure or access the appropriate functionality), these users are likely to share their accounts, too [202]. (We consider related problems in a PKI, namely credential delegation in Section 5.5.1 and Chapter 7.) Plus, their motivation to adhere to security policies they do not understand is low in the light of a lax overall security culture at their company [4].

2.2.2 Internet Security

In this section we describe how phishing, faked information, and users' lack of knowledge concerning SSL negatively affect Internet Security. These three issues are closely related as attackers often combine several of them.

2.2.2.1 Phishing

Identity theft on the Internet has become a very prominent attack those days as neither technology nor users are prepared to fight it accordingly. It is estimated that two million users in the U.S. have been the victim of identity theft in 2004 [239]. This attack comes in the form of so-called *phishing* in the online world. The term phishing stems from the conjunction "password fishing" and is reminiscent of the slang word "phreak" (phone freak). Phishing typically works as follows. At first, a large list of target email addresses is assembled. This task is quite simple and cheap as 100.000 valid email addresses cost only a few Euros on the SPAM black market [184]. The investment quickly pays if there is only one victim in this set. It is estimated that an astonishing 5% of recipients respond to scam email [12]. About 2% of people surveyed by Gartner Group reported that they had lost money after being "phished" [239].

In the second step, email is sent to the target addresses which purports to come from a well-known company, service etc. the addressee already has a business relation with. The email asks the recipient to update his or her profile stored by the service or to take a similar action that requires

prior user authentication. Figure 2.5 shows an email, which was received by a friend of the author and fooled him at first sight. The hyperlinks

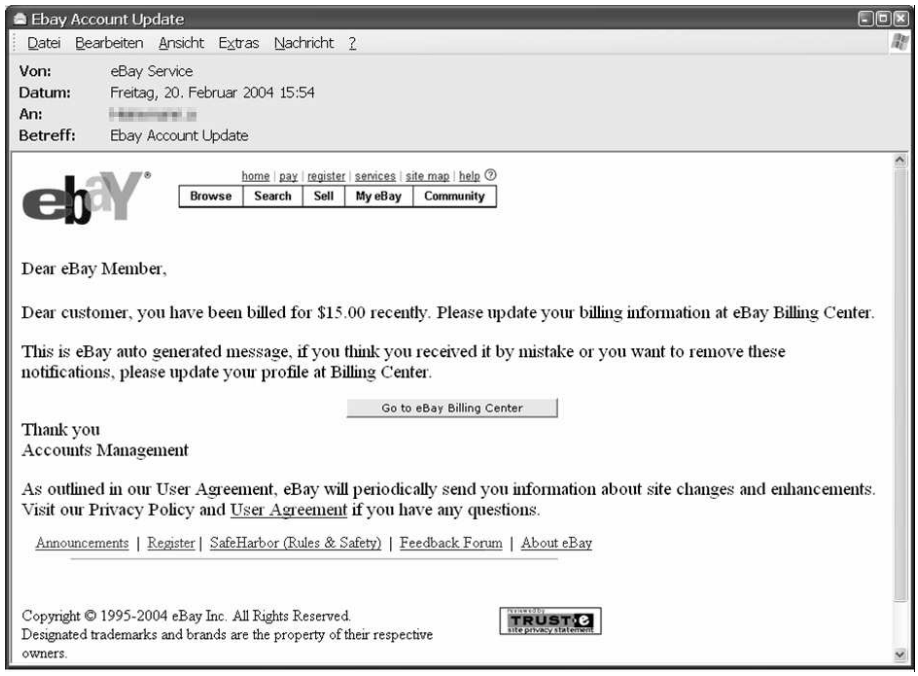

Figure 2.5: Phishing email purportedly coming from eBay.

contained in the email are all referring to genuine eBay web pages except the one that reads "Go to eBay Billing Center". This link leads to a web site which was set up to steal the personal data of incautious users. The site perfectly adheres to eBay's look & feel and is therefore pretty hard to identify as a fake. What may make people suspicious is the fact that it requests plenty much of personal data. Among others, name and address, eBay user ID, bank account number, debit card number (including the 3-digit security code on the back), social security number (SSN), and even the debit card PIN used in ATMs are asked for. Obviously, this is enough to rob the victims' bank account by cloning ATM cards [156], mess with their eBay accounts, or abuse their SSN (which is often used as a secret, e.g. for password-recovery at other web sites).

Phishers typically use compromised web servers in order to stay untraceable in cyberspace. The attackers' web pages are online only for a short time (a week or so on average) until the hijacked brand gets to know about it and forces the respective ISP to close it down. However, this time frame is often

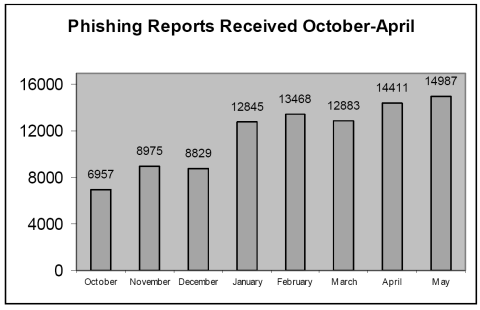

Figure 2.6: Phishing statistics October 2004–May 2005 (source: [112]).

long enough to mount successful attacks and gather customers' data. In December 2004 attackers exploited a server misconfiguration and set up spoofed *PayPal* web pages at the author's web site. With over 50 million customers, eBay-owned company PayPal is the leading provider for electronic payments on the Net. As such, PayPal has constantly been the target of phishing attacks [216]. In the concrete case, users were referenced to the spoofed web page and were asked to enter personal and account information. A script was loaded on the server to sent out the information to an email account. After PayPal had learnt about the fraudulent web site it urged the author's ISP by email to shut down the web site.

Phishing has become a major plague for the companies that are affected. This is the reason why the method of sending out information to customers by email is completely discredited. A recent MailFrontier[4] survey on phishing has revealed that 28% of test subjects were duped by phishing email. Plus, 20% of the participants regarded genuine email confirming a purchase as fake. This provides strong evidence that commercial email is getting a more and more unrealiable medium with respect to authenticity. Interestingly, the vast majority of companies or individuals does not use a technology suitable to provide authenticity – digitally signed email. We come back to this point in Chapter 6.

2.2.2.2 Faked Information

Another kind of attack is the dissemination of incorrect information, e.g. in order to influence stock markets. The following example and screenshot are drawn from [44]. On April 4th, 2003 Asian stock markets went into a

[4] `http://www.mailfrontier.com`

spin due to the announcement that Bill Gates allegedly was killed in Los Angeles. However, this spectacular news item was in fact forged. According to the announcement the founder of Microsoft Corporation and richest man in the world died in the aftermath of an assault at a Los Angeles charity event.

Figure 2.7: Faked web site purportedly coming from CNN.

Stock brokers were the first to circulate an email referring to a web site which appeared to come from the news broadcasting station CNN. The fraudulent page (see Figure 2.7) imitated CNN's design which made people

believe that it was authentic. Even some TV stations fell for it. As the page address is `http://www.cnn.com@cgrom.com/news/law/gatesmurder/index.shtml` people are likely to think it is in fact a page located at `www.cnn.com`. They walked into the attacker's trap since the true location of the page is `cgrom.com/news/law/gatesmurder/index.shtml`. The trick consists of abusing a standard-compliant, but rarely applied mechanism to convey a user name and password to a web site that requires HTTP Basic or Digest Authentication (see Section 7.2.1). The URL (uniform resource locator) syntax `user:passwd@host` or `user@host` is a shortcut to login using the credential (`user`, `passwd`) to the page at `host`. The information shown on the page was deemed authentic since not only the visual design was a good imitation of CNN's, but also the URL seemed to point to CNN. Since a lot of people around the world regard CNN as the source of trustworthy information, the attack succeeded – which would probably not have been the case if the URL was simply `http://cgrom.com/news/law/gatesmurder/index.shtml`.

The attack vector used in this example was an exploit on humans' perception and capability to make an informed security decision. Only few of the average Internet users know about the particular syntax for an URL, so they need the support of their web browser to see what is going on when accessing such a page. URLs often involve a list of `&`-separated parameters for scripts run on the web server or long numeric session identifiers etc. Therefore an unreadable URL containing special characters like the colon, the at, or a percent sign and a long string of encoded data will not surprise users. As Table 2.2 shows, browser manufacturers have, unfortunately, not yet agreed on a uniform reaction of web browsers to this threat.

Clients like Opera or Firefox display warning messages when opening the page. However Opera's address bar afterwards shows the URL as it was entered except that a "password" parameter is substituted with "*****" if present. It is surprising that, albeit the trick has been known for a considerable time, Internet Explorer (IE) or Mozilla do not inform the user or support her in another way (e.g. by ignoring the syntax completely). At least IE changes the URL to the actual site (the screenshot in Figure 2.7 was taken while the page was loading). Firefox displays an alert (see Figure 2.8), but does not unescape special characters which may confuse the

Browser	Warning	Address Bar
Internet Explorer 6.0	–	`user:password@` prefix stripped off and true domain shown
Opera 7.54	√	password hidden if present, remaining URL unchanged
Mozilla 1.6	–	URL unchanged, shown with complete `user:password@` prefix
Firefox 1.0	√	`user:password@` prefix stripped off and true domain shown

Table 2.2: Web browsers' reaction on URLs containing the @ sign.

Figure 2.8: Alert shown by Firefox for @ URL.

user. Anyway phishers can easily avoid such an alert by setting up web pages that do in fact require authentication.

Countermeasures against faked web sites as well as phishing attacks have been proposed. *Netcraft* for instance offers a free anti-phishing toolbar that integrates itself into Internet Explorer or Firefox (see Figure 2.9). The toolbar shows how long the current web site has been online, its rank based on Netcraft's web server statistics, and the country where it is located. An overall "risk rating" is computed out of these parameters. Our examples shows the web page located at `www.paypai.de` – a URL which is easily confused with `www.paypal.de` as in some fonts the upper case "I" resembles the lower case "L". The *eBay Toolbar*[5] also protects eBay or PayPal customers from inadvertent password disclosure. Herzberg and Gbara's *TrustBar* is a similar browser extension that uses logos to tag known and trustworthy web sites (see [123] and `trustbar.mozdev.org`). In cases where SSL is used, a logo of the certification authority which has issued the site certificate is also

[5] `http://pages.ebay.com/ebay_toolbar/index.html`

shown. However, we argue in Section 3.2.3 that knowing the issuing CA does not say much about the security level of the certificate.

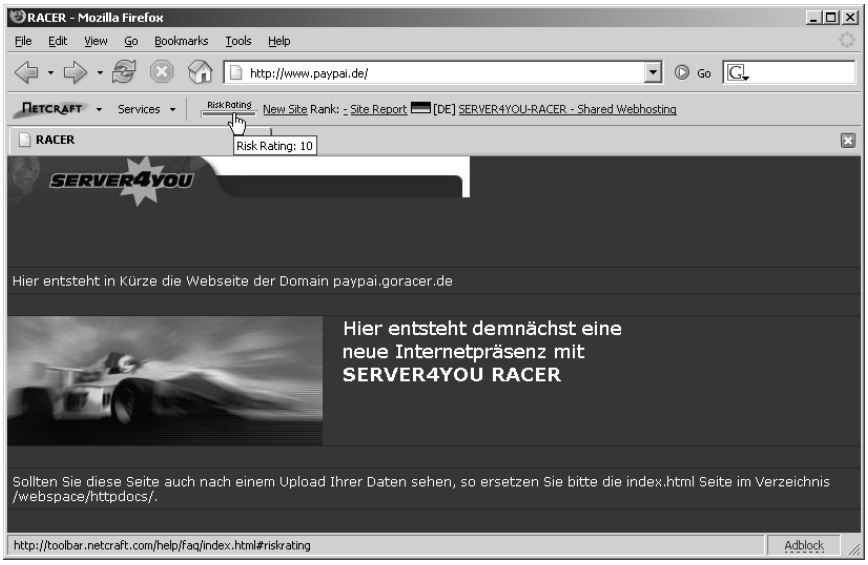

Figure 2.9: Web page `www.paypai.de` viewed with Netcraft Toolbar.

2.2.2.3 SSL

In the last subsection on Internet Security we look at SSL as the prevalent security mechanism on the World Wide Web (WWW). This protocol is typically used to establish a secure channel between the client and the server that is encrypted, integrity-protected, and (unilaterally) authenticated at the same time. Albeit SSL offers strong authentication (i.e. using public key cryptography), client-side authentication is mostly done by user name/password credentials.

A very typical scenario is where customers provide credit card and bank account numbers or other sensitive private data to a commercial web site. The first important requirement is to ensure that the browser is talking to a genuine server which belongs to the company. The second requirement is that the data is kept secret from third parties while in transit (maybe on a local network, the ISP's network, or intermediate machines on the Internet etc.).

Friedman *et al.* [88] studied users' capabilities and knowledge towards web security. Users were asked to explain their mental model of a "secure

connection" and how in practice they distinguish secure (i.e. SSL) connections from insecure ones. During the test the subjects were shown actual browser screenshots. The participants were more successful in correctly identifying non-secure than secure connections, but still one third of them was not able to recognize non-secure pages. Participants based their decision on different pieces of evidence. Most of them looked for a key symbol or an icon of a closed padlock as most browsers use these metaphors to indicate a secure connection (cf. Section 6.5.3 for the results of our own user study for secure email). However, the error rate was quite high. Web sites are free to include pictures of padlocks, seals[6], and labels that appear trustworthy where in fact they are not. Therefore, just looking for such an icon "somewhere on the screen" does not help. Interestingly, the metaphor of a padlock was often found to conceptually mislead users as a padlock may suggest that one's data is stored at a protected place instead of being transferred in a secure way.

The second most important evidence was the content of the web site. This indicator, however, has proven to be very error-prone as we also saw before. Only a minority (20%) of test participants correctly used the protocol name at the beginning of the URL as an indicator for secure connections (`https://` instead of `http://`).

Cutting a long story short, a significant portion of users has difficulties in applying standard web security mechanisms properly. As a consequence they resort to their own "ad-hoc" or "rule of thumb" security rules. And note that the user test did not even take into consideration the confusion (i.e. warning messages) users are faced when connecting to an SSL web site with a certificate that the system cannot validate because of an unknown issuer or a subject name/domain name mismatch. As average users only have a vague understanding of "security" [88, 83], they cannot precisely name and enforce the protection goals of confidentiality and authenticity. Web browsers do not present comprehensive information to fulfil the task of "manual" certificate validation reliably either [228].

[6]For instance, the certification authority *Thawte* (`http://www.thawte.com`) uses so-called site-seals which indicate the validity of SSL server certificates. Thawte customers may include such seals on their web page to increase their trustworthiness and provide a shortcut to a certificate validation page on Thawte's site. However, Gutmann [118] remarked that the mechanism does not work reliably. And what is even worse: web site visitors do not seem to care much about an invalid seal.

Figure 2.10: SSL warning dialogue in Internet Explorer.

An inadvertent case study confirmed the thesis that users are likely to dismiss and ignore SSL warnings [119]. The responsible persons of a New Zealand-based bank forgot to renew an SSL certificate for their web server before it expired. As a consequence, the bank's customers were temporarily exposed to the corresponding alert windows of their web browser. The astonishing fact is that during this unintentional usability test, 299 out of 300 (!) candidates ignored the warning on the first page and continued to login in [23]. It is very likely that a large part of them would have also continued when presented a certificate with an unknown issuer as Internet Explorer's UI does not make a big visual distinction between the different types of errors (which could have very different security meanings). A screenshot of the dialogue that appears in case one of the three conditions fails is shown in Figure 2.10.

According to the 2004 Netcraft survey on SSL usage [171], there are about 1.7 million pages on the Internet that use SSL (see Figure 2.11). But only 15 % of them present a certificate that the user's browser successfully validates (i.e. with no alert windows coming up). About one in eight certificates were found to have expired or to be not yet valid. Another 35% were found to not match the domain name, e.g. when connecting to `www.ba.com`, a certificate for `www.britishairways.com` is presented. For Internet Explorer users this yields the same alert window with a slightly

different message (the third of the conditions shown in Figure 2.10 fails and gets marked with a triangle instead of a hook). Such negligence, which yields to "false positive" error messages, is critical as the binding between domain name and certificate subject name is the mechanism to parry MITM attacks (the recently published [121] attacks on the TAN system applied by online banks would succeed if users ignore such a warning). The remaining 38% of certificates are self-signed or issued by an authority whose certificate is not a pre-configured trust anchor in common browsers. In the light of the results of the user studies mentioned above one may wonder how users react in such situations. Evaluating and measuring user behaviour would be an interesting task for further user studies.

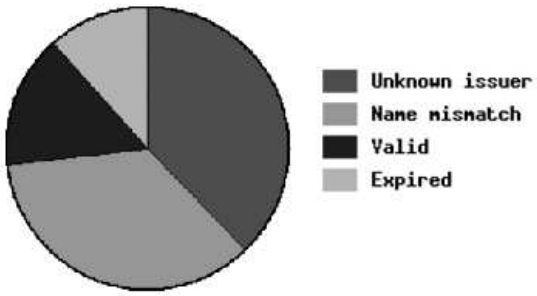

Figure 2.11: Netcraft 2004 SSL survey (see [171], 1,745,795 web sites examined in total).

2.2.3 Cryptographically Secured Email

As we mentioned before, standard email is insecure by design. Security was not an explicit criterion in the RFC 822 specification [58] written in 1982. But today, email is exposed to a bunch of threats: messages can be intercepted or modified rather easily while en route, e.g. at occasions where people use public WLAN hotspots, which is becoming more and more popular. Email that cannot be delivered to the intended recipient (e.g. because of a typo in the address or an overfull mail box) usually bounces back or gets forwarded to a system managers account. Travelling two or more times through the net, the message is again exposed to the risk of eavesdropping.

Apart from phishing, the numerous ongoing virus attacks are another important issue. Computer viruses and worms try to access the user's ad-

dress book on an infected machine in order to spread further. Worms like *Sober* reproduce themselves by sending out copies with their own SMTP engine. The nasty thing they can do is to forge the `From:` argument of the email. Either by inserting an arbitrary address or an address found in the affected user's address book as a sender. They disguise the true location of the infection, making it hard to inform the right user to run antiviral software. Plus, recipients of infected emails are less sceptical if the sender address is familiar to them and are therefore more likely to click on the attachment. For the time being, people with a non-English mother tongue are protected to a certain extent as an English email from a friend makes them suspicious. However, it is very likely that viruses will soon evolve further and become context-aware, e.g. by adjusting their language according to the top level domain of the email address. This development can currently be seen with phishing emails.

Public key cryptography could do a great job in fighting these threats as it provides sender authentication and message confidentiality. However, secure (i.e. encrypted and/or signed) email is rarely found on the Internet today albeit the technology is widely available. The first version of the software package PGP was written nearly 15 years ago. It (or an open-source derivate) is available for free on every major platform (Linux, Mac OS, Unix, Windows etc.) at least in the form of an external crypto application working via the clipboard or as a plug-in for mail user agents (MUA). On the other hand, virtually all popular contemporary MUAs such as Microsoft's *Outlook/Outlook Express*, *Netscape Mail*, *Mozilla Mail*, *Thunderbird*, *Pegasus Mail*, or *The Bat!* have built-in support for S/MIME[7] [44]. Still the relative user population both for PGP and S/MIME is infinitely small [117].

One might argue that user interest is low due to the current business model for certificates that calculates costs on the potential benefit instead of their real practical value [7]. For S/MIME to be useful, an end user has to get a certificate from a well-known CA (Certification Authority). Otherwise, recipients of signed email will not have the root certificate preconfigured as a trust anchor in their MUA. This makes it hard for them to verify the signature and – even worse – they are "punished" by the additional costs to read through security messages and act accordingly. For instance, the window shown by Outlook Express when faced with a signature that

[7] S/MIME stands for secure MIME, i.e. Multipurpose Internet Mail Extension.

cannot be validated is a powerful deterrent [44, 94]. As a consequence an end user has to pay an annual fee which typically exceeds the benefits of the certificate. (Cf. [155] for a thorough discussion of business models for CAs.) Business issues, however, should not serve as an excuse since more experienced users might find out that there are zero-cost email certificates for S/MIME issued by CAs whose root certificate is widely deployed (e.g. Thawte). PGP certificates are for free anyway due to the PGP trust model. Nevertheless, there remains a significant cognitive overhead of certificate management which is in the focus of Chapter 3 and Chapter 6.

Usability issues in the context of secure email were first studied by Whitten and Tygar in their seminal paper [253]. They conducted a user test where participants were asked to process standard tasks with PGP 5.0. For instance encryption and digital signing, key retrieval from a communicant or from a key server. The PGP version under test had a visually appealing graphical user interface (GUI) and ran on the Apple Macintosh platform which is usually considered a shining example for a user-friendly interface. Yet test results were disappointing as two thirds of the participants did not manage to sign and encrypt messages properly with PGP. A majority of users did not grasp the concept of public key cryptography as it was presented to them by the PGP user interface. As a consequence, they used their own or a third persons public key to encrypt outgoing messages by mistake. Plus, the participants showed a poor understanding of the idea of certification (or "key signing" in PGP jargon). Only some users realized that they should care about key authenticity. None of the test participants used PGP's complex key and trust management facilities to establish trust in a public key they retrieved from the key server or got unauthenticated by email.

In another email security-related usability evaluation Gerd tom Markotten [98] assessed a plug-in for Microsoft Outlook. This plug-in was certified for the usage with *qualified digital signatures*, which should provide a high level of security (see e.g. [15] and page 62 for details). The test of the plug-in of the German *Signtrust CA* revealed an astonishing number of 90 security-critical usability problems. A significant amount of the problems found is due to unintelligible and technical language which makes it very difficult for security novices to use the Signtrust Mail plug-in properly and comfortably.

2.2.4 Online Privacy

Gerd tom Markotten [98] also uncovered a number of usability problems with JAP[8] (Java Anonymity and Privacy). JAP is an HTTP proxy that allows users to surf the Web anonymously. JAP supports cascades of independent "mixes" – an idea originally due to Chaum [53]. The system also protects against eavesdropping on the user's local network as the communication channel is encrypted from the client machine to the last mix in the cascade. An early version of JAP was criticized for its lack of appropriate user feedback and the difficulties to configure it properly. *Tor*[9] is a similar approach to JAP, but in stark contrast, Tor does not have a user interface at all, making it – as its developer argues [68] – very usable. Such a radical design choice is surely reasonable in situations where users might mis-configure the tool and open up security holes. However, an application that works completely transparent and does not offer any feedback about the security state violates the design principle of self-descriptiveness (see Table 2.3). As a consequence, users might become uncertain as they cannot easily and reliably tell whether the tool works or not and they might all the more breach security.

Similar observations were made in an evaluation of the filesharing application KaZaa [110]. Filesharing is very popular these days as people can exchange music, videos, and software with others for free. KaZaa operates in peer-to-peer mode, i.e. all computers in the KaZaa network are by and large equal in that they operate both a server (offering information) and client (downloading it). Peers can offer files on their local hard disk for "sharing" with others. Local information of a peer is automatically indexed and made available on the whole network such that others can download it comfortably. It has turned out that a majority of test users have difficulties in configuring their local shares properly, i.e. to restrict sharing to certain directories and not inadvertently grant access to the complete hard disk. KaZaa users were also found to not understand that sharing is a priori not restricted to multi-media files only.

[8] `http://www.inf.tu-dresden.de/~hf2/anon/`
[9] `http://tor.eff.org`

2.3 Multi-Layer Classification of Methodologies

The contribution of this section is a novel systematics of previous work and existing methodologies in the field of usable security. We identify a total number of six aspects, which we can arrange in a multi-layered model (see Figure 2.12). Along the vertical axis, a distinction is made between general, abstract proposals and more concrete and practical ones. It is possible to interpret the "lower" layers of our model as those farer away from the user and the "upper" layers as those closer to her. From Chapter 3 of this thesis onward, we present a number of new approaches. We have depicted their logical position in the figure.

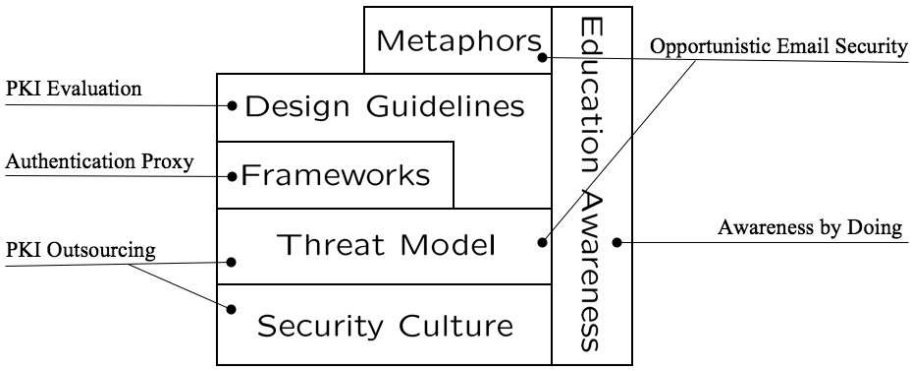

Figure 2.12: Multi-layer classification of usable security.

Which Layer is the Right One? All of those layers have their justification for a secure systems designer, but – depending on the actual use case – usable security may be realized more effectively on a particular layer than on the others. It is therefore crucial to identify the right layer for a given problem. As they affect the user interface or interaction mechanisms, changes on the upper layers are more likely to be technically easier to realize. However, innovations on the lower levels are more likely to have a fundamental impact – making applications more usable. As a simple example consider how secure email was realized. In order to let users encrypt email, the metaphor of a sealed letter came up, materializing as a graphical icon in most MUAs while the corresponding cryptographic functionality was retrofitted. What the designers had in mind was the idea of end-to-end security, which is laudable yet pointless. As we all know, only a very small portion of email is

encrypted or signed, secure email did not succeed on a larger scale. Now, let us make some mental acrobatics and imagine that email and Internet service providers would have set up an infrastructure allowing them to secure *all* mail travelling between them. Suddenly, the fraction of secure email would be near 100 %, simply by changing the threat model ("all others are evil") and a cultural leap of faith ("I rely on my ISP to care for me").

Structure For each layer, we discuss current approaches or projects illustrating the respective ideas. We start our analysis with the most general issue, namely the layer "security culture" which describes the way we think about security (Section 2.3.1). A natural point to continue is the discussion of threat models which affect the design of security mechanisms. The lack of usable security software is widely acknowledged. To improve this situation, engineering frameworks and interface and interaction design guidelines have been proposed (Sections 2.3.3 and 2.3.4). Frameworks and design guidelines are drawn in a way which expresses a certain entanglement. Consider for instance the approach of user-centred security engineering (see below) where both areas go hand in hand. The topmost layer is formed by metaphors and the mental images users have when working with secure applications. As the user's conceptual model is not part of the design, but results from the system images, this layer is a bit smaller in the drawing. Adjoined to it, we can find education as it can directly influence these images. Education is drawn vertically as it affects all of the layers (think of the training of system developers). Awareness and motivation is a precondition for successful training. Therefore awareness and education are summarized in a vertical layer (Section 2.3.6).

2.3.1 Security Culture

Some authors have analysed the relation between security and safety in order to see what concepts could be transferred from one area into the other, we summarize their findings here. The role of security outsourcing has not yet been discussed from a usability perspective. Delegating complicated tasks to a server or an organization is an appealing solution to solve (or rather: shift) usability problems. However, the question is justified if outsourcing does not conflict with protection goals. A whole chapter of this thesis is

dedicated to the outsourcing of trustcenter capabilities, so we do not treat this subject further here.

Security and Safety In everyday language, the terms "security" and "safety" are often used synonymously, for instance the German translation for both words is "Sicherheit". In a technical sense, they have different meanings, albeit both areas are linked together. "Security" is typically defined by referring to defence measures that ensure a state of inviolability from hostile acts or influences while "safety" focusses on the protection against accidents and (mechanical or human) failure which may cause harm to the system. Several authors have pointed out that it could be beneficial for the IT security community to look at how safety techniques are successfully applied in practice [39, 200].

There are obvious reasons to do so: Both safety and security are secondary goals which exist "to protect an organization and its staff while they are engaged in the primary task – production" (Brostoff *et al.* [39]). There are a number of other similarities between the domains of security and safety: An economic tension between these requirements and the production tasks exists as both sides are competing for resources (cf. discussion at the beginning of this chapter). Compared to the production tasks, the outcome of safety and security measures cannot be predicted due to the stochastic nature of incidents. Plus, the feedback associated with those measures is largely negative and compelling only after an incident (cf. the discussion about awareness in Chapter 4). The domains also resemble in the way problems are handled. Such problems are often attributed to the carelessness or incompetence of employees and are being followed by punitive management actions. The domains, however, differ in society's attitude towards breaches. While violations of safety rules are mostly considered negative, this is not the case for security mechanisms. For instance, hacking is idealized or working around security mechanisms is seen as a "badge of seniority" [4]. Awareness campaigns are an important way to change this view.

Fail-Safe Systems The aerospace sector was cited as a shining example for its high safety standards including the rigorous software development process and maintenance regime, the oversight by regulatory bodies, the demanding training for pilots, and the intense investigation of failures [200].

This community has done its homework in refraining from the general attitude to blame accidents solely on human error. However, this opinion was prevalent two decades ago and significantly hindered the development of safety-critical systems [39]. Immediate adoption of aerospace standards to IT systems is not likely to happen on a broader scale. It is a lame comparison as avionics software is no general consumer software and one cannot expect that only skilled and trained personnel will operate security-critical systems. A more realistic role model might be that of the automobile industry as Sandhu [200] suggests. Cars are in fact a consumer product deployed in large numbers and driving a car requires a "small, but non-zero amount" of previous training.

The security community none the less has already taken up the idea of a "fail-safe" system. Such systems are designed to not cause harm when they fail (and crash "gracefully" so to say). An example is Maseberg's fail-safe concept for PKIs [161]. Here the possible "failure" means cryptographic algorithms that may be broken or become weak over time. This is currently the case with cryptographic hash functions (see e.g. [247, 248]). A fail-safe system design anticipates the potential incident from the outset and provides appropriate countermeasures (for instance in Maseberg's scenario for fail-safe signatures, data is signed multiply with independent algorithm suites). Some ideas of the safety community have found their way into secure applications, e.g. in the form of the "path of least resistance" design guideline recommending fail-safe default settings (see below).

2.3.2 Threat Model

A very general yet important distinction, system designers should always bear in mind is the difference between the *theoretical* (or ideal) and the *effective* security level of a system [69]. In the following we have a close look at this phenomenon. We believe that an appropriate threat model is crucial for the usability of secure applications. Often enough, security mechanisms are deployed with only a vague understanding or schematic idea of what the security threats are. (On the other hand it may be necessary to sometimes regard a system's regular users as potential attackers – this is covered by the concept of multilateral security, an example of mobile commerce can be found in [237].) This results in over-sized solutions that are too hard to handle. We give some examples below.

Theoretically Secure is Not Effectively Secure Secure systems are typically designed according to a specific threat model that is formulated in terms of security goals similar to those described in Section 2.1.3. For instance, during the design process some pieces of data are classified as confidential which implies that they should leave the system only with an appropriate level of protection (i.e. encrypted under the recipients's public key using one of a set of certain symmetric ciphers with prescribed minimal key lengths etc.). The design process assures that the theoretical level of security is very high, even against sophisticated cryptographic attacks.

Now compare this to the effective level of security which is achieved when the human factor comes into play: Assume that the user in question does not manage to import the recipient's public key into the system properly (because she is incapable of finding the right menu, determining the correct certificate directory settings, or understanding the trust model etc. to name only a few possible hurdles). Furthermore, we assume that she still wants to send the document. Then it is not unlikely that she will find a way to do so without encryption. With regard to the design objective "confidentiality", the effective level of security in this small example is non-existent, although the system may offer cutting-edge, military-strength encryption in the ideal case.

To give a more concrete example we consider the use of S/MIME to encrypt emails. It is a common, yet incomprehensible dogma that an outgoing email can only be encrypted when the sender also has her own key pair. The technical reason for this constraint is the idea to store the message encrypted in the "sent" folder on the hard disk. There are several flaws in this concept: Firstly, the threat model does apparently not only take into account attackers that eavesdropped messages while en route, the mechanism also restricts access to the victim's messages on the hard disk. However, file encryption is not what S/MIME is originally for. Secondly, the previous statement only holds for messages that were sent out in encrypted form – why not for all the others? It would be no problem to simply encrypt the mailbox file as a whole. Thirdly, why does the mechanism require the sender to have a certificate, why not simply use password-based encryption for protecting the session keys? (When stored in software, the private key is protected by a password anyway.)

The "all-or-nothing" way S/MIME is implemented in MUAs may be well-meant, but has the consequence that messages are sent in the clear which could have been encrypted – a result which somehow reduces this particular threat model to absurdity. Note, that we did not yet speak of the case when a communicant's certificate cannot be validated. Similarly, the message is sent in the clear, readable for everyone, instead of encrypting it with an "unknown" and not yet validated key, which would reduce the number of potential attackers to only one. We further elaborate the subject of opportunistic behaviour in security in Chapter 6.

Less is More The idea is slowly catching on that finding a reasonable trade-off between security and usability is superior to aiming at perfect security in vain. Smetters and Grinter [215] are rather straightforward in this respect, and ask "if you put usability first, how much security can you get?". Their concept is very similar to the pragmatic approach of Sandhu [200] who states three golden rules of "good-enough security":

1. Good enough is good enough.
2. Good enough always beats perfect.
3. The really hard part is determining what is good enough.

It is of particular interest where complex security measures cannot (due to technical reasons) or should not (as not to restrict user convenience) be strictly enforced every time (cf. our Figures 2.2 and 2.3). Both engineers and users may have difficulties to understand the functioning and limitations of a certain involved technology or complex policy. Therefore, weaker techniques can surpass such technologies in practice when people are more likely to "know what they are doing" and how to behave correctly. Secure systems should try to meet user's expectations and mental models about security instead of telling them that their expectations are wrong [111].

Being Paranoid Does Harm The enforcement of password policies is a typical example. We focus on one aspect of password policies to illustrate the problem: A lot of system administrators take it as an obligation of users to change their login password regularly. Modern operating systems like Windows XP have built-in mechanisms to enforce this rule in fixed intervals of e.g. 30 or 60 days. However, a significant part of users feels uncomfortable

with it and tries to work around it [4, 204]. In order to prevent users from re-entering their current (or a previously used) password when it comes to a mandatory password change, systems keep a password history for each user – this was called some kind of "stalking" [202]. While in this situation, a lot of effort was put into technology, the actual threat model was not stated explicitly and precisely. From a security point of view, not much is gained if passwords are re-newed every month instead of every year, say. If one has a very powerful attacker in mind that mounts an offline attack on the encrypted password file, this precaution reduces the attacker's time frame only by a small factor. In contrast, the enforcement of password *quality* or entropy would have a much greater impact on security. Password entropy depends on password length, the size of the alphabet password characters are picked from, and – above all – the way how passwords are generated (see Chapter 4 for details).

Online guessing attacks can be detected and defeated no matter how often the password is changed. Protection against shoulder-surfers might be an argument for the strict change policy. Shoulder-surfing refers to an attacker who tries to learn the password by surreptitiously observing a user entering the password on the keyboard. Consequently, one might say that the strength of a password decreases with every time it is used. However, the 30-days policy exactly does not address this issue since the number of logins during the respective period is not taken into account. On the other hand, a lot of memorability problems arise with passwords that are used infrequently and that are subject to a strict change policy [204].

If the threat model really emphasizes shoulder-surfing, other and more appropriate mechanisms are at hand, e.g. the method of Roth *et al.* [192]. If the threat model emphasizes password guessing attacks, the rule to never write down a password should be abandoned as robust passwords are too complex to remember – an explicit recommendation by security expert Bruce Schneier [3]. These considerations again show the importance of analysing and stating a matching threat model.

Being Pragmatic Pays Off User identification and authentication on the Internet is a good example where pragmatic security mechanisms have been widely deployed. If you would ask – in an abstract way – a security expert how "to secure an application responsible for business transactions

on an insecure network", you would probably get the answer to use a PKI for user identification and public key cryptography for authentication. However, this is absolutely not the way things happen on the Internet today. The overwhelming majority of electronic commerce web sites does not rely on a PKI to identify their customers. (Even major players like *Amazon* do not strictly enforce an SSL-only access policy as this could exclude some customers.) In fact, fraud, phishing, and identity theft do happen on the WWW as we can see in Section 2.2.2, but the pressure to adopt a more secure technology is not yet there as financial losses are still too small.

The common security mechanism of email-based identification and authentication (EBIA) relies on email addresses as unique identifiers and the ability to receive email at a certain address as an authenticator. This mechanism is often used for password resets. In this case, the new, machine-generated, password is sent out to the user or a hyperlink that leads to a page where a new password can be entered (the link contains a unique number as an authenticator). Obviously, the security relies on the assumption that it is hard for an adversary to read a user's mailbox – an assumption that has proven to be reasonable in most practical cases. We refer the reader to [90] for a thorough discussion of the potential as well as the limitations of EBIA in practice. A major benefit of this approach is the fact that users (both end users and system engineers) can assess the security properties of the system, which clearly are not very high, but have proved to be robust in the users' own experience.

EBIA is a good example for a reasonable threat model that led to a fairly usable and secure mechanism. A "textbook" threat model of a powerful attacker would probably have resulted in a more complicated mechanism as we argued above. We believe that the "good-enough" and "put usability first" approach mark a profound change in the way many "secure" systems will be engineered in the future. Rather than being driven by a conservative assessment of threats and security requirements, the safety and usability of such systems will be at the focus of many low to medium risk application areas.

2.3.3 Engineering Process Frameworks

Numerous frameworks exist for general software engineering as well as security or usability engineering in particular. There are basically three ways

of how to get a usable security application [260]. Two of them are naive solutions which take as a starting point a usable (resp. a secure) application and add security (resp. usability) to it. We discuss their drawbacks and advertise the approach of user-centred security as an alternative.

Adding Usability to a Secure Application We have already sketched usability methods in Section 2.1.4. All those can be applied in a security context as well. Eckert describes a number of security engineering methods in her book [70]. These methods often fall back upon general software development process models (see e.g. [19]). The four steps of plan-build-assess-modify form the basic phase of the process which is iterated multiple times. This process can be adopted to the security setting as well. A functional analysis of the future system forms the starting point. Security issues are then addressed by means of a threat and risk analysis. Here the impact of a predetermined set of attacks towards the system is assessed such that a security strategy can be formulated. The subsequent design and implementation phases take turn with testing and evaluation periods.

From a software engineer's point of view, adding usability to an application whose security mechanisms are fixed is problematic. In this case applications tend to make technical details directly visible without a further level of abstraction [97]. Abstraction on the UI includes, for instance, the use of meaningful metaphors or the design of interaction mechanisms that separate security from primary tasks. Such a lack can be seen for instance in the field of email clients. Consider for instance the key and trust management in PGP: Attempts to facilitate and improve the UI without touching the underlying concepts only addressed the symptoms, but did not change things fundamentally. The approach of Dhamija *et al.* [64] focussed on making visible PGP's complex internal trust metrics and the way the authenticity and trustworthiness of a key is calculated. However, their interface ended up with technical details that only have marginal importance for the productive task of sending secure email. Whitten [251, 252] took a somehow opposite approach in forcing the user into training lessons before letting her or him access key management functionality. This can be a last resort, but surely not a panacea.

Adding Security to a Usable Application Another approach is to take an existing usable application and add security features (Schneier [206] iron-

ically calls this "sprinkling on magic security dust"). The field of Computer-Supported Cooperative Work (CSCW) is a typical example where this strategy has been followed [260]. An important drawback of this approach is the fact that it tends to produce "security-agnostic" applications as security functionality is kept transparent or separate from the core functionality, for instance when security features of the underlying operating system (OS) are used. This approach has its pros and cons. On the one hand, application designers relying on OS security capabilities do not have to re-implement functionality, which anyway is a tedious and error-prone process, possibly going beyond their knowledge. On the other hand, contextual information, which can be useful when a security-relevant decision is to be made, is usually only available in the application itself [215]. As an example consider the situation where an incoming email message carries a digital signature that cannot be verified as no appropriate certificate path can be established. If the email is a contract, the recipient will care about its authenticity, but not if the email's content is unimportant. The current email standard RFC 822 provides only a rudimentary way to include meta-information about the content of an email.[10]

To cut a long story short, some security-relevant decisions will always have to take into account contextual information that is why they cannot be completely left to lower technical layers. Nevertheless, some tasks can and should be handed over to the OS or a crypto library for the sake of security (and a hopefully standard-compliant and proper implementation). Functionality that is both essential from a security point of view and invisible to the user typically is suitable to be delegated to the operating system, like the generation or protection of keys or the provision of good pseudo random numbers. Tasks that require a direct user involvement should ideally be provided by some kind of toolkit that can be more tightly integrated into an application. Smetters and Grinter [215] call this the "Lego Bricks for Security" approach.

User-centred Security The third and most promising approach is a synthesis which considers both aspects simultaneously from the outset. Zurko and Simon [260] have come up with the notion of *user-centred security* to

[10]The header field `X-Importance` stores a numerical value representing the (of course subjective) urgency of the email.

"refer to security models, mechanisms, systems, and software that have usability as a primary motivation or goal". Concrete engineering guidelines have been formulated by several authors (see next section). In the remainder of this section we have a closer look at engineering frameworks that integrate security and usability from the beginning. *AEGIS* (Appropriate and Effective Guidance for Information Security) is a high-level framework which is due to [81]. It is a systematic approach based on the spiral model of software development where multiple specification-implementation-testing phases alternate. AEGIS starts with asset identification, which means the specification of security requirements. During a risk analysis and security design phase, vulnerabilities are identified. Costs and likelihood of a certain attack are taken as an indicator which countermeasure to select. Each countermeasure's costs are then weighted against the impact of the attack to make an appropriate choice. Gerd tom Markotten [97] synthesizes an established security engineering method with best practices of usability engineering. This highly technical process is combined with a method due to Nielsen [173] to obtain her user-centred security engineering (*UCSec*) method. Nielsen's usability engineering process is characterized by parallel design and testing phases. UCSec pays special attention to user behaviour (as users may be both the subject or object of an attack), user needs, and sets forth a number of general usability guidelines for security software (see Table 2.3).

Without doubt, the latter of the three approaches is the most promising one. There is consent that neither security nor usability can be simply "retrofitted" after completion of the design process [259, 17]. However, as secure systems only change slowly, there are often situations in practice where user-friendly mechanisms had to be retrofitted as the original design did not anticipate them. As an example consider our approach to the problem of credential delegation presented in Chapter 7. Here the constraint is that technology on the server side is completely fixed and the client side should not significantly change as well.

2.3.4 Design Guidelines

In the previous section, we confined ourselves to general, high-level frameworks governing the development process as a whole. Compared to these frameworks, which represent a planning strategy and organizational structuring, engineering *guidelines*, which we will discuss now, provide more con-

crete support for developers who are about to solve a concrete problem or make a design choice. There is a range of recommendations, most of them are stated in an ad-hoc fashion centred around sample applications or experience reports. Saltzer and Schroeder [199] advocate for a system design as simple and small as possible with mechanisms that have fail-safe defaults, and are easy to understand and use. Although their proposals date back to the year 1974, their general principles are still valid today. Apart from the two usefulness-related rules *economy of mechanism* and *psychological acceptability*, e.g. the *least privilege* or the *complete mediation* principle have made their way into modern operating systems [70].

General guidelines for the design of ergonomic user interfaces have been stated in the ISO standard 9241 [131]. They are listed in the first column of Table 2.3. Gerd tom Markotten and Kaiser [100] rightly argue that those guidelines cannot be used literally in a security setting. Their modified guidelines are shown in italics in the right column of the table. We have added references to the five properties listed in Figure 2.4 in order to motivate each adaptation.

Implicit Security A lot of usability problems with secure applications originate from the logical and/or technical separation of productive and security tasks. An elegant way to overcome this problem is to let applications infer a user's security objectives (the secondary goal) from her productive tasks (the primary goal) and the corresponding UI actions and inputs. This approach was termed *implicit security* [215]. Associated with this challenge is the feedback about the current security state such that users are able to control and change the state without hassle. Dourish and Redmiles [69] suggest using event monitoring combined with security heuristics (comparable to those in intrusion detection systems) to visualize the security status. Their approach aims at providing users extensive information about configuration, activity, and important parameters of a networked system.

In independent work, Yee [258, 259] uses a similar approach to incorporate security decisions into the users' workflow. Yee proposes to initially grant only minimal abilities to applications and to enhance these authorities through explicit user interactions. An important benefit of the idea that user interactions can convey security implications is a strategy called *security by designation*. The set of permitted and acceptable actions stays coherent

general principle	**relevance for security applications***
error tolerance	*Error prevention* is considered the most important principle because of P4 and P5.
suitability for individualisation, controllability	Users should receive *more rigorous guidance* where needed to avoid weaknesses due to misconfiguration or mistakes (P4).
suitability for learning	Humans tend towards the "trial and error" method when starting to use an application (P1). Even a carefree *first-time operation must not lead to security breaches* or severe errors.
self-descriptiveness	*Presenting status or security messages clearly and precisely* is crucial (P2, P3).
conformity to user expectations, consistency	The application should use a correct, homogeneous and usual *terminology* to describe things (P3).
suitability for the task	Users want to accomplish security-related tasks easily and efficiently. The design should take into account that users have varying abilities and do *not consider security as a primary goal* (P1).

* P1, ..., P5 refer to the properties listed in Figure 2.4 on page 21.

Table 2.3: UI design principles for security applications (based on [100] and [233]).

with what the user actually wants, but this does not require confirmation prompts or an a priori formulation of a policy. Security by designation is user-friendly as decisions can be made in a meaningful context (cf. the discussion on page 46). Yee gives the examples of file access rights or cookie management. In contrast, the strategy that most applications follow is *security by admonition*, i.e. to explain to users what may happen when they try some unacceptable action and demand their confirmation (for example: "After deleting your private key, you won't be able to read emails encrypted with that key – do you want to proceed?"). Yee's guidelines are built around the concept of actors (applications, other users of the system etc.) and their abilities (those of interest for the current user). Granting authority should always require a user action which conveys it implicitly or explicitly. The UI should make abilities visible and provide methods to revoke them.

Yee's approach has some limitations as it is more or less tailored to the management of privileges like access to resources (file system, address book, user interface, network connection etc.) and for applications running

locally on the desktop. Up to now, it deliberately ignores the task of user authentication in an open network, which accounts for a large portion of usability problems with PKI-enabled applications [253]. An extension in this direction is an open research problem.

2.3.5 Metaphors

Logical and/or visual metaphors are methods to make the operation of a complex technique easier for users by mapping analogies of their everyday life to technology. When it comes to concrete user interface design of a secure application, the challenge is to find understandable, precise, and appealing metaphors for security-related tasks, objects, and properties. The padlock icon, which is used in most web browsers to indicate SSL connections, is an example for a metaphor that is widely applied. This does not necessarily imply that the metaphor is precise or cannot be misunderstood. Friedman *et al.* [88] report that a padlock may give users an incorrect idea about SSL (see Section 2.2.2.3). However, it seems that users regard the padlock icon in general as a natural choice to represent the concept "security" (see [193] and Section 6.5.3). In contrast to web browsers, a uniform labelling mechanism for digitally signed email is not implemented by current MUAs [92].

Mental Models Metaphors are closely related to mental models as they provide shorthands for communication with the system. Mental models refer to the conceptual understandings that users hold of the domains in which they operate; actions are planned and interpreted with respect to these models [69]. A mental model can be formed through experience, training, and instruction. It depends on the interpretation of perceived actions of a system and its visible structure [179]. Ideally the user's model matches the designers' conceptualization of the system. In her PGP usability test Whitten showed that this condition is not always satisfied: many participants of the test "did not understand the necessary conceptual model of public key cryptography well enough to be able to figure out which public key to use when encrypting a message for a particular recipient." [253, 251]. It was proposed to visually represent key pairs according to the *Yin-Yang* symbol, where two individual halves coloured black and white fit together, but this did not lead to an improvement in users understanding of the concepts of public key cryptography. This provides some evidence that complex

PKI-enabled applications cannot be made significantly easier to use without touching the technical layer.

2.3.6 User Education and Awareness

At the end of the day, most security mechanisms require some form of user interaction to be effective. Consider for instance the TCP/IP network stack as an example. Data encryption and integrity protection with the help of public key cryptography may take place on the data link, the network, the session, or the application layer. L2TP (Layer 2 Tunnelling Protocol), IPSec, SSL, and PGP are examples for each alternative. Albeit these technologies are quite different, all of them nevertheless require a manual initial configuration and continuous management (e.g. for the update of trust anchors). The term "user" surely extends from end user to system or network administrator in this case. At this point, security depends on the correct behaviour of the responsible person. We repeat once again the observation that security is a secondary goal for users[11], which directly decreases user's motivation to adhere to security policies or to take pains in operating the application correctly.

User Attitude With a lack of motivation and awareness users tend to underestimate the risk of being the target or victim of a security incident. "I'm not so important" is a common statement [250], obviously ignoring the fact that, for instance, breaking into an arbitrary user account on a system allows an attacker to get one foot in the door of an organization and possibly gain administrator privileges afterwards. A similar attitude is "I have nothing to hide" [138]; such employees do not overlook the far-reaching consequences of their carelessness. People were also found to violate security because of personal attitude ("I am not pedantic/paranoid", [250]) or social issues ("Sharing a password is a sign of trust", [204]). Furthermore, it was shown that – due to a lack of security knowledge and information about real dangers – users construct their own, inadequate mental model and are guided by vague experiences [204].

Creating and maintaining information security awareness is therefore an obvious goal. However, awareness and motivation cannot be created by drill

[11] With the sole exception of "security service providers", for instance ISPs that offer VPN services for their customers.

or threat of punishment, which is a frequent implicit assumption among the management, possibly another relic from the time before security escaped the military environment [4]. An individual chapter of this thesis is devoted to security awareness, so we do not explore this issue further at this point.

User education is surely another important building block. It is consent that user education is a good thing in general and for secure applications in particular. In the light of the disappointing results of her usability study, Whitten proposes to pay more attention to user education and guide users in a step-by-step fashion through the application [252]. She suggests a concept of "safe staging", which increases the application's complexity that is visible according to the current knowledge and experience of the user. A weaker form of this concept is contained in the idea of "user levels", where tasks are automated with a wizard for inexperienced users and experts are allowed to tweak detail settings.

Limitations Usability expert Jakob Nielsen [175] has recently voted against relying too much on user education. In his view, this does not resolve the general problem of removing burden from the user, which originates from preconditions and assumptions on the technical layer. User education can be a last resort for existing secure applications with poor usability that are mission-critical (e.g. for military purposes). There should also be user training for secure applications as it should be for any other type of software in an enterprise setting. However, one should not expect a home user having received previous training or being willing to do so. We came to the same conclusion at the end of Section 2.1.4. Schneier [206] sums up the importance of education and awareness for enterprise security in saying "Security measures that aren't understood by and agreed to by everyone don't work".

2.4 Conclusions

In this chapter we recalled the common terminology from the area of usable security like threats, protection mechanisms, and the like. As an introduction to the subject, we studied a number of security incidents and examples in order to derive a categorization of usability-related security problems. We introduced a fine-grained distinction between user-caused violations, shaded by the degree of intent, versus application-caused vulnerabilities, shaded by the degree of impossibility. Finally, we turned to the question of how the

usability of secure applications can be improved. The main result of this chapter is a multi-layered model consisting of different hierarchical "layers" which describe approaches to tackle the problem. The pros and cons of a design choice in favour of a certain layer are discussed. We also reviewed related work and classified it according to our model. For each layer, we present an example in the remainder of this thesis to demonstrate the respective methodology. We believe that, in the context of PKI, especially the threat model should deserve more attention as various reasons for usability problems originate from this layer. This may lead to technical innovations that – by design – evade major problems from the outset.

Chapter 3

Measuring Usability and Utility of PKI-enabled Applications

> *Q: How is a key pair like a hand grenade?*
> *A: You get two parts, there's no aiming, and it's hard to use safely.*
>
> *Q: How are they different?*
> *A: With a grenade, you throw the dangerous part away...*
>
> – Don Davis

> *If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.*
>
> – Bruce Schneier

In Chapter 2, we followed the well-known approach to IT security in that we identified security threats and formulated the corresponding protection goals. We now show how these goals can be achieved in practice by cryptography-based and PKI-based mechanisms. This chapter starts with a brief introduction into these mechanisms before a number of technological challenges are outlined (Section 3.1 and 3.2). Being aware of such challenges is crucial as they directly affect the usability and may hinder the deployment of PKIs as well.

It is obvious that the UI plays the major role for usability. But in the end it strongly reflects and depends on the underlying concepts of the software, especially for secure applications as we have already argued in Section 2.3.3. We will see that – compared to well-known security mechanisms like passwords or file access control – public key cryptography and PKI are a much more complex matter, less intuitive, and more difficult to understand for users (and software engineers as well). Looking behind the scenes is also instructive as PKI users will get involved in technical details sooner or later. Davis [60] points out this as an inherent property and the price to pay for the advantages of PKI. He observes that systems using public key methods burden the user with responsibility for tasks that are otherwise being centrally managed by an administrator or a server.

The main topic of this chapter is our framework to measure the usability and utility of PKI-enabled applications. This framework grew out of a large usability study [44], which was conducted by the Darmstädter Zentrum für IT-Sicherheit on behalf of Microsoft Deutschland GmbH and was co-authored by the author of this thesis. The scope, the testing process, and the findings of this study are summarized in Section 3.4.

The framework has subsequently been applied to the evaluation of trustcenter software, namely the Windows 2003 Server CA and Entrust's PKI package (see Section 3.5 and [150]).

3.1 PKI in a Nutshell

PKI provides an infrastructure – the "I" in PKI – for the use of cryptography in large and dynamic user groups. In Chapter 2 we emphasized openness as central property.

The approach we will follow now is bottom-up in that we first explain the necessary basics of cryptography before we delve into the details of PKI. We treat the subject straight-forward as we start with the idea of encryption and the notion of symmetric cryptosystems, then we proceed to asymmetric cryptography by a discussion of key management issues. Furthermore, we pay attention to the problem of public key distribution and show how to provide integrity and authenticity. Digital signatures, certificates, and the concept of a public key infrastructure plus some practical considerations conclude this section.

We can only give a coarse introduction here, a more comprehensive treatment is out of the scope of this thesis. The reader is therefore referred to the textbooks [8, 42, 70, 163, 205, 219, 224] to name but a few. Readers familiar with the subject may want to proceed directly to Section 3.2.

3.1.1 Confidentiality through Encryption

The need for confidentiality in personal or organizational communications dates back to ancient times. Gaius Iulius Caesar is probably the most popular and most often cited historical example in crypto literature. He already gave a precise description of an elementary *cipher* in his book [47] two thousand years ago. It transforms a *clear text* (or *plain text*) message by substituting letters one by one. Clear text letters are circularly shifted three positions such that – when applied to the (modern) Latin alphabet – A, B, C, ..., W, X, Y, Z, thus get mapped to D, E, F, ..., Z, A, B, C. The process of transforming clear text into *cipher text*, that is unintelligible for an outsider, is called *encryption*. The inverse operation is called *decryption*. The *Caesar cipher* is a simple, so-called *substitution cipher*.

Obviously, some kind of secret information must be part of every cryptosystem to provide the legitimate communicants with an advantage compared to an attacker from the outside. In the example above, the encryption algorithm (the idea of shifting letters) is such a secret piece of information, while the number of shift positions (which is 3 in the example above) is the other. According to Kerckhoff's principle stemming from the 19th century [141], the communicants should not rely on the assumption that the former piece of information is unknown to the attacker. For their security, they should not rely on the secrecy of the cipher algorithm, but only on its variable input data, which is called a (cryptographic) *key*.

Of course, for the simple substitution cipher with a key consisting of a number between 1 and 25, the "security margin" between the communicants and the attacker is rather small. Modern cryptosystems have a much larger key space, which effectively inhibits an attacker from trying out each possible key in order to decrypt a cipher text. Typical state-of-the-art cryptosystems have key spaces consisting of for instance $2^{128}, 2^{192}$, and 2^{256} (AES) or 2^{112} and 2^{168} (3DES) elements – the time for a complete test would exceed the lifetime of the universe even on a very powerful computer (see [27] for some illustrative figures). A cryptosystem is considered strong enough if no attack

is known which takes less effort than on the order 2^{80} operations [163]. This gives a lower bound for the size of the key space.

The approach of Kerckhoff is justified as revealing the design of the algorithms allows independent experts and the scientific community to scrutinize the cryptosystem's security properties and to rule out design weaknesses. Besides, keeping the algorithm of a cryptosystem itself secret is virtually impossible if the system is used by a large number of people and en-/decryption devices might get lost, e.g. in the army while in action.

A system which has undergone intensive studies and which is widely deployed is – with a high probability – stronger than a "home-made" or uncommon scheme.[1] It is therefore reasonable that applications only support a selection of established algorithms. Leaving too much choices to a user in this respect can be dangerous (cf. Section 3.3.4.1).

3.1.2 Symmetric Cryptosystems

Albeit not stated explicitly, we have exclusively spoken of cryptosystems so far that are *symmetric* in that the keys used for encryption and decryption are either identical or the latter can be derived easily from the former (e.g. a 3-step encryption shift corresponds to a decryption shift by $26 - 3 = 23$ positions in the Caesar scheme).

Up to the mid 1970s only symmetric ciphers had been known. We explain the idea of asymmetric ciphers in Section 3.1.3. Asymmetric ciphers are very important as they help overcome some drawbacks of symmetric ciphers and open up a number of new applications within "asymmetric cryptography".

The Key Management Problem Consider a group of persons where each two of them want to communicate in a confidential way over an insecure network such that the rest of the group cannot understand the data they exchange. This implies that each pair has to use its own secret key. Several problems arise in this setting if a symmetric cipher is used. At first, each two communicants have to securely agree on a common secret key prior to their actual communication. This requires what is called an "out-of-band"

[1] Even if a weaknesses of a widely used algorithm is found and exploited by an attacker, such news will spread quickly. For instance, during World War II, the Allies were able to read German radio messages encrypted by the Enigma, but could not make tactical use of them as this would have informed the enemy that the system is broken.

channel. Such a channel might be a personal contact or a secure second communication medium (telephone, letter, etc.). Pairwise secure out-of-band key exchange may be feasible in small user groups. However, it does not scale well as the number of keys, which need to be securely generated, exchanged, and stored, totals $\frac{n(n-1)}{2}$, n being the number of participants. The quadratic complexity renders this scheme impractical, since even in a community of only 1000 users, nearly half a million keys are required.

Apart from this fact, the necessity for an initial out-of-band channel also precludes people to directly communicate with previously unknown peers in a secure way – a feature we identified as crucial for open networks. In such a situation, the only way to avoid sending messages in the clear (apart from not sending them at all) is to rely on a third party which is trusted by both communicants. Acting as an intermediary the third party would receive messages encrypted with the sender's key, decrypt, and re-encrypt them with the recipient's key. This has the obvious drawback that all messages must be routed through and can be read by the intermediary. As a bit less security-critical alternative, the third party could act as an "introducer", which generates a new secret key and transfers it securely to both parties. We will soon see that there is also an introducer in the figurative sense in the context of asymmetric ciphers, but not with opportunistic security (Chapter 6), which can be considered a main characteristic. Such a *key distribution center* (KDC) is, for instance, an integral part in the famous Kerberos authentication and key distribution service [221, 145]. In a Kerberos "realm" (i.e. an network domain or organization), each user has a bilateral trust relationship to a central entity that operates the KDC. Kerberos also allows the interconnection of different realms (e.g. crossing the borders of a company) thus supporting the requirement of openness to a certain extent. However, individual users who are not member of such a realm, cannot communicate securely with others.

However, a centralized KDC suffers from the typical problems all centralized IT systems suffer: Without further technical measures, it represents a single point of failure and a single point of attack. Plus, it may also become a communication bottleneck in large user groups [8].

3.1.3 Public Key Cryptosystems

The aforementioned problems of pairwise secret key exchange and scalability can be overcome by asymmetric cryptography. The ground-breaking idea of Diffie and Hellman to construct ciphers where the encryption key and decryption key are significantly different such that computing the latter out of the former is infeasible in practice. The main characteristic of an asymmetric cipher is the use of a so-called *key pair* consisting of a *public key* and a *private key*. The public key is the encryption key while the private key is the decryption key, so both operations can be separated. Here encryption is often called a *trapdoor one-way function* as it is infeasible to invert it (one-way property) without the knowledge of the private key (trapdoor information) [163].

At the time their seminal paper [67] was originally published in 1976, no instance of such an *asymmetric cipher* was known. The RSA cryptosystem (named after Rivest, Shamir, and Adleman who came up with the idea in 1978, [191]) is still the asymmetric cipher used most often in practice. Mathematical details of RSA and its variants are given in Chapter 5. Typical key lengths for RSA are 1024, 2048, or 4096 bit. For practical purposes, asymmetric ciphers are in general not used directly to encrypt payload data as they are significantly slower than symmetric cryptosystems. Instead, payload data is first encrypted with a symmetric session key (or transaction key) generated at random. This key in turn is encrypted with the recipient's public key.

3.1.4 Authenticity and Integrity Protection

The computational asymmetry between the encryption and decryption operation in an asymmetric cryptosystem can be exploited in a second way. Consider how a paper document is signed with a handwritten signature – a process that is typical for the everyday business world and that is regarded as being reasonably secure. Such a signature provides authenticity of the data towards an arbitrary party as forging a handwritten signature is deemed rather difficult while it can be easily verified (e.g. by means of the reference template contained in an ID card). A *digital (or electronic) signature scheme* has very similar properties, but is even harder to forge. Like an asymmetric cipher it makes use of a key pair where the private key is required for the creation of the signature while the public key alone allows

its verification. In the RSA signature scheme for instance, the operation for signing (verifying) data uses the same maths as the operation of decrypting (encrypting) in the RSA cryptosystem.

Hash Functions Signature schemes do usually not work directly with the data to be signed, but apply an initial transformation that shortens the input to a few bytes. Such mappings are called *cryptographic hash functions*. They assign a bit string of fixed length (128, 160, 224, or 256 bit are typical values) to a bit string of arbitrary length. This has the advantage that the length of the signature always equals the key length. (Compare this to the case of encryption where the cipher and the plain text have equal length[2].) In particular it does not depend on the length of the document itself that is why the document cannot be reconstructed from the output value alone (the scheme is said to not support *message recovery*).

A hash function h is suitable for digital signature if it is *collision resistant*, i.e. the chance of finding two different messages m, m' with $h(m) = h(m')$ is negligible. (Such collisions nevertheless always exit as h is not injective.) Collision-resistance is a must as the signature is tied to the hash value of the message, not the message itself. Otherwise, the signer could later deny having signed m and claim that he signed $m' \neq m$ if $h(m) = h(m')$. This violates the requirement of digital signatures to be non-repudiable. (The trick works the other way round when Mallory sells something to Bob and lets him sign the order m which maps to the same hash value as a text m' with a different price or amount of goods.)

A weaker requirement than collision resistance is second pre-image resistance (also called weak collision resistance), which means that, given a certain message, it is infeasible to find a different message such that both messages map to the same hash value. This property prevents fraud as a digital signature does not directly refer to the document, but only indirectly via the hash value.

Non-Repudiability The non-repudiation feature is a precondition for digital signatures to be useful for legally binding declarations of intention, e.g. for business transactions, contract agreement, etc. Signers should be prevented from being able to repudiate a signature they created with their

[2]Not taking into account data compression of the clear text before encryption or padding.

private key. Given the strength of the underlying digital signature scheme, the only plausible reason to do so is that their private key got lost or became broken. Therefore, EU legislation dictates that cryptographic devices like SmartCards, which had to undergo a security evaluation, should be used to store the private key. Such devices apply the private key for signature computation, but do not reveal it to the outside.

According to the EU directive 1999/93/EG [181], "electronic signatures" may have three different legal qualities as an evidence. Only for two of those levels, namely advanced and qualified signatures, digital signatures in a cryptographic sense are compulsory. The main differences between those two levels is the fact that the latter one requires the private key to exist only on a SmartCard and that the binding of the key pair to an identity is confirmed by an appropriate authority. German laws allow to use (with a few exceptions only) qualified signatures in place of handwritten signatures [144]. An equipment for qualified signatures (consisting of a SmartCard, an approved card reader and appropriate software) and the service of certificate renewal still costs a significant amount of money compared with the possible benefits for individual users. For instance, people buy goods online without caring about digital signatures and they only have one or two situations a year where digital signatures may be useful for the communication with public administration. Benefits are thus more on the administration's or on enterprise users' side [155]. Besides, the effective legal value of an advanced, but non-qualified, signature is not yet clear.

Message Authentication Codes Non-repudiability is currently a rare requirement in practice. On the other hand, as we have already pointed out in Section 2.1.3, repudiability indeed might be a desirable feature. Cryptographic authentication protocols should also avoid hidden challenge semantics, i.e. messages that are determined completely by one party and signed by the other one. For example, this was a privacy issue with an earlier proposal for an authentication protocol for electronic passports [236]. A more profane reason not to use asymmetric crypto is performance. Hash-based message authenticity codes (MACs) consist – roughly speaking – of a hash function depending on a secret key. They can only be used between parties that agree on a common secret in a separate way.

3.1.5 Public Key Distribution

Since knowledge of the public encryption (resp. signature verification) key does not allow to derive the private decryption (resp. signature) key, the former can be made publicly available for everyone without affecting security. A way to publish such information could for instance be a public directory similar to a phone book (as already proposed by Diffie and Hellman), a personal or corporate Web page, the footer of an email, or a business card. (Users of the PGP system often use one or more of these three possibilities to disseminate their public key or at least its "fingerprint" or "thumbprint", i.e. a hash value, and a reference to the whole key, which can then be downloaded from a public *key server* – see below.)

The fact that public keys can be made public allows to solve the key exchange problem elegantly. Given the electronic equivalent of a telephone book that stores the public encryption keys instead of the phone number of each user, confidential communication is possible even in an open user community and especially without prior out-of-band key exchange. In this setting, the implicit assumption is made that public keys retrieved from the directory are authentic, i.e. the lookup process returns the genuine public key of a certain entity. This point is crucial as Mallory, the attacker, may try to make Bob believe that a public key belongs to his communicant Alice when in fact it belongs to Mallory. As a consequence, Bob might be tricked into inadvertently encrypting a message for Alice with Mallory's public key. (A similar attack is possible in the case of signature verification keys.)

There are basically three, significantly different, ways of ensuring a public key's authenticity. Each of these variants has its pros and cons. Although the third variant is the one used most often, the other two also have their justification in practice.

(i) The simplest approach is for Alice and Bob to exchange their public keys bilaterally and in an out-of-band fashion. This guarantees a high level of confidence when Alice and Bob know each other or authenticate with each other using official paper documents, e.g. a passport or company badge. However, this scheme suffers from nearly the same weaknesses as pairwise out-of-band secret key exchange outlined in Section 3.1.2, namely practicability and scalability, with the exception that the communication must not be confidential, but authentic in the present case.

(ii) An appealing idea is to rely on the trustworthiness of the source of information, i.e. Bob would always assume that the key repository returns the genuine key attributed to the person Bob asked for. From a user's viewpoint, this reduces the problem to ensuring that the repository cannot be tampered with (i.e. the information's integrity is preserved), the assumption that the repository as a third party works in due form, and the communication channel back from the repository to the requesting user applies some form of authentication. The first and the second requirement are out of a user's scope. Compared to variant (i), the authenticity of an entire electronic repository can usually not be verified directly (while this works for a printed phone book, which an attacker is unlikely to forge).[3]

A similar mechanism is used for the dissemination of public keys, which are pre-installed in operating systems (shipped on a CD), applications (downloaded from a repudiable Web site), and the like.

(iii) A drawback of the second variant is the fact that the authentication only covers the source of information, but not the data itself. With entity authentication, however, the data retrieved from the directory is only useful for the requesting party, but cannot be verified by others (including the party itself at a later point in time). This limitation can be overcome by letting a trusted party issue "digital IDs" that bind an identity to a public key and that are digitally signed by the issuer – comparable to the analogue of a paper ID card. The idea of *public key certificates* (or digital certificates) is due to Kohnfelder [146]. Certificates form the basis for an infrastructure that provides the authenticity of public keys (a public key infrastructure or PKI for short).

3.1.6 Digital Certificates and Public Key Infrastructures

In the previous section, we motivated the need for verifying the authenticity of public keys and explained different mechanisms to do so.

In a *narrower sense*, a public key infrastructure can be seen as a security infrastructure. Such an infrastructure lets participants communicate with

[3] In fact, the Bundesnetzagentur as the German authority for a digital signature infrastructure uses a similar mechanism. The public keys retrieved from the "white list" directory are authentic per definitionem.

each other securely using public key cryptography and allows them to verify each others keys with the help of certificates and trust relations. A formal calculus of trust in the authenticity of public keys is due to Maurer [162]. In this model participants can compute a numeric value indicating how much confidence they have in a certain public key (based on their own, incomplete view of the whole set of certificates). The model captures the idea of certificates as well as the notion of direct trust (first variant in 3.1.5) and indirect trust ("I trust you to vouch for another person's key" – variant (ii) or (iii) in 3.1.5).

In a *broader sense*, the term PKI also captures the technical infrastructure behind the certificate and trust infrastructure. This view, taken for instance by [8], comprises a number of concrete services (like issuance, storage, and revocation of certificates, the generation, backup, and recovery of keys, time stamping and so on) and logical components (like certification authority, registration authority, directories etc.). This perspective leads us to the notion of a trustcenter, i.e. a trustworthy organization that delivers one or more of those services to its customers. We have a closer look on the internal processes of a trustcenter in Section 5.1.2.

It will become clear from the context whether we speak of a PKI in the narrower or broader sense in the following. When speaking of PKI, some people implicitly mean an X.509-based infrastructure or assume a particular (the "centralized" or "hierarchical") trust model. Both constraints probably hold for the majority of actual PKIs (in the broader sense). Although most of the existing PKI-enabled applications are X.509-based, we do not a priori exclude others as the analysis and results of this thesis also apply to them.

ID Card Analogon We have already introduced the concept of a public key certificate in the preceding section and linked it to ID cards and the like. Certificates are similar in that they contain a unique serial number, the name of the issuer and the subject (i.e. key holder), their affiliation and/or location, a validity period, possible usage clarifications ("for all countries", "for signatures and financial transactions up to 10.000 EUR" etc.). Certificates are digitally signed by the issuer. As a consequence, if Alice obtains a certificate purporting to be Bob's and signed by Trent, Alice has to obtain Trent's authentic signature verification key, then check the integrity of his signature on the certificate, and then trust Trent that he in fact diligently

verified Bob's identity before issuing the certificate (like it should be the case with the issuance of ID cards).

Certificate Chains As we can see, the problem shifts to ensuring the authenticity of Trent's public key. Trent may in turn have a certificate for his public key signed by Trudy, then Trudy's signature has to be verified and so on. Such a series of certificates where the issuer of certificate i is the subject of certificate $i+1$ is called a *certificate chain*. Obviously, such chains cannot continue ad infinitum. The authenticity of the public key of the last entity in the chain can only be established by one of the other means listed in Section 3.1.5. Finally, this always boils down to the non-technical issue of trust. The last public key in the chain is thus called a *trust anchor*.

Trust Models A lot of research effort has been put on the role of trust in the areas of cryptography and security (see [162] for an overview). Different trust models have evolved that differ in the entities that are assumed as trustworthy.

Very often the distinction between the centralized and the decentralized trust model is made alongside the different certificate standards, namely X.509 [52] and PGP/OpenPGP [48]. This is short-sighted as the centralized model can also be implemented with PGP certificates. On the other hand, a decentralized approach would technically be possible with X.509 certificates, but X.509-based clients only support the centralized approach.

In a centralized model only a few dedicated players issue certificates to a large number of subjects. Such parties are called *certification authorities* (CAs) in contrast to *end entities* that cannot issue certificates. They might be official institutions like public authorities, commercial or non-profit organizations like banks or universities respectively. The amount of trust relying parties place in such CAs strongly depends on their overall reputation and the security of their certification process. As such, professional CAs take pains to provide a high level of security working with skilled personnel, evaluated and certified hard- and software, extensive physical security measures etc. Such installations are therefore also called *trustcenters*. CAs often form hierarchies, e.g. VeriSign has one so-called topmost or "root CA" and several subordinate CAs for different purposes. We also speak of root certificates omitting the CA in between.

The so-called *web-of-trust* or decentralized model is a generalization as all players can issue certificates, no matter whether they are individuals or large organizations. This property makes up the main difference to a purely CA-based PKI. As a consequence, there is typically more than one certificate chain for a certain public key, so public keys can "accumulate" trust. Each user has the freedom to judge the authenticity of a public key based on the number of disjoint chains, their lengths, and the trust placed in the intermediaries.

Personal Security Environment The storage location of an individual's private key is called a *personal security environment* (PSE). Private keys can be stored in a so-called software or hardware PSE (also called softtoken and hardtoken). Depending on the enrolment process, PSEs are produced in a trustcenter or by the certificate holder himself. We also use the term credential when speaking of private keys, which indicates their use for strong authentication.

A software PSE is a piece of data on the file system that can be protected by a password (or another key), for instance according to the PKCS#8 and PKCS#12 standards [195, 196]. If the PKCS#12 format is used to protect a private key when send from the trustcenter to an end entity, one also uses the term "transport PIN" or "transport password". In order to access the private key inside a softtoken, the user has to provide a password. Depending on the application, the key may or may not be cached in memory until it is used the next time. One also uses the term *key store* (or key ring) to denote a file that contains private keys as well as other parties' certificates in a confidential and integrity-protected way respectively. Apart from PKCS#12, Java key stores or the containers of Microsoft's CryptoAPI for instance provide the same functionality.

Obviously, softtokens have some shortcomings from a security perspective. For instance, they can be copied easily when stored on a network drive. Hardtokens are devices like SmartCards (requiring an additional reader) or USB sticks (ready to plug in without additional hardware). Hardtokens are designed to implement a two-factor authentication (possession of the token and knowledge of a PIN) and to make the theft and/or illegitimate use of private keys nearly impossible. An important characteristic of hardtokens is the fact that they do not hand out the private key to the application, but

compute the necessary operations on their own. However, this does not completely prevent fraud as a signature application may show a document on the screen while letting the hardtoken sign the hash value of another. This is called the *presentation problem* as USB sticks or card readers typically do not have the ability to display the whole document to be signed [70].

Revocation Management PSEs might get lost or destroyed. As a security precaution, PKIs allow to *revoke* certificates of end entities or even CAs when the private key is suspected or known to be in danger or unusable. We describe the way of how this situation is handled in an X.509-based PKI. A *certificate revocation list* (CRL) according to the X.509 standard contains the serial number, the revocation date, and optionally the revocation reason. It is digitally signed by the CA that issued the certificate or a proxy thereof. In the latter case, one uses the term "indirect CRL". CRLs are updated on a regular basis, say once per month, and immediately if a certificate is revoked. CRLs are usually distributed according to the "pull model" which means that clients have to retrieve the information from the CA. OCSP [168] is a client/server protocol that, in contrast to CRLs, provides real-time information about the status of a particular certificate. Support for CRLs and/or OCSP is crucial when certificates have a long validity (for instance, some of the VeriSign certificates that are shipped with Windows XP have a 30 years lifetime).

3.2 Challenges

In the previous section, we laid out the main concepts of cryptography and PKI. We did this mostly without referring to their concrete realization or actual implementations, which we will catch up on now. This section is entitled "challenges" as some of the tasks necessary to put PKI into practice require elaborate technical mechanisms or even fall into the category "left as an exercise" we introduced in Section 2.1.4. Therefore applications which fail to address these challenges suffer from usability problems. Among the tasks discussed here are the lookup and validation of certificates, the secure and reliable management of credentials and the interpretation of policies and trust models. These issues were taken into consideration for the organization of the evaluation framework in Section 3.3. This thesis proposes new solutions to some of the problems in Chapter 5 to 7; the corresponding

references are given below. In what follows we often have an X.509-based PKI in mind (especially when referring to technical details), but the observations hold for PGP-based PKIs in a very similar way. Plus, looking at other technologies gives us further insights about innovative approaches.

We remark that the evolving technologies of *identity-based encryption* and *identity-based signatures* (IBE, IBS, see [210, 33]) is no panacea to the challenges listed below. IBE offers no end-to-end security as it requires trusted third parties, the so-called private key generators (PKG). In order to make use of IBE, the encrypting party has to know the parameters of the PKG (an issue of information access as described in Section 3.2.4) and check their authenticity (similar to obtaining an authentic root certificate, see Section 3.2.2).

3.2.1 Certificate Validation

Let us first look at certificate validation, which is a frequent task in a PKI. Certificate validation occurs each time the authenticity of a public key is checked before the key is applied to verify a signature or to encrypt data. The following aspects have to be taken care of. The presentation is guided by the one in [212], an algorithmic and much more comprehensive description can be found in RFC 3280 [128]. The complexity of the RFC-compliant validation is illustrated, e.g. by the flow chart depicted in [43], Muñoz *et al.* describe certificate validation as "one of the toughest scalability problems in a PKI" [167]. Due to this high complexity the process might exhaust the resources of, for instance, mobile devices. Proposals have been made to allow the delegation of one or more steps (like lookup, signature verification etc.) to dedicated parties, so-called validation authorities (see e.g. RFC 3029 [6] or the New Security Infrastructure NSI proposal by Fraunhofer-Institute SIT [129]). Unless not explicitly stated otherwise, we restrict ourselves to the so-called *shell validity model*.

Certificate Chain Building The correctness of the signature of a certificate is checked by applying the issuing CA's public key. In order to check in turn the authenticity of this key, the whole certificate chain has to be processed. PKI-enabled applications build the certificate path automatically based on their locally available information (contained in their certificate store and possibly retrieved online). This is not an issue users

have to cope with as long as the certificates of the entire chain are at hand, valid, and trusted. In case the public key of the last certificate in the chain is untrusted or an intermediate certificate is invalid (for one of the reasons listed below), this also requires a positive, manual security decision about the trust anchor, an intermediate CA, or the end entity certificate itself.

Validity Period The currency for each certificate in the chain is also checked. In the shell model this requires that the current point of time (i.e. the time when the validation algorithm is executed) lies in the validity interval defined by the certificate. Otherwise the applications treat the certificate in question always as invalid – although this does obviously not imply that the key is insecure. At first sight, this constraint seems rather simple to meet and enforce.

However, we know of no PKI-enabled application which automatically ensures that the clock on the end user's computer is up-to-date, e.g. via the Network Time Protocol [166]. For instance, Firefox at least reminds the user to check that the local time is correct, but Microsoft applications simply state that "the certificate has expired or is not yet valid" (they all behave likewise as they share the same operating system libraries for certificate and key handling).

On the other hand, there are often no mechanisms established to guarantee the automatic renewal of certificates before they expire. We already mentioned in Section 2.2.2.3 that, unfortunately, one in eight SSL web pages presents a certificate chain to web browsers where one or more certificates are not yet or no more valid. Among those are the web sites of banks or for instance of a security conference[4] – i.e. sites maintained by people who really should know better. Although this does not necessarily mean a security breach, as we argued above, it is very critical from a usability point of view: As SSL warnings appear so often for no good reason, users get conditioned to ignore them generally no matter what the respective situation is like.

Certificate Extensions and Semantics Version 3 of the X.509 standard allows to include *extensions* in a certificate. While there are only 16 so-called *standard* extensions (plus two more defined by the PKIX working group of the IETF in [128]), each CA in turn is free to define its own, *pri-*

[4] `https://amsl-smb.cs.uni-magdeburg.de/sicherheit2006/`

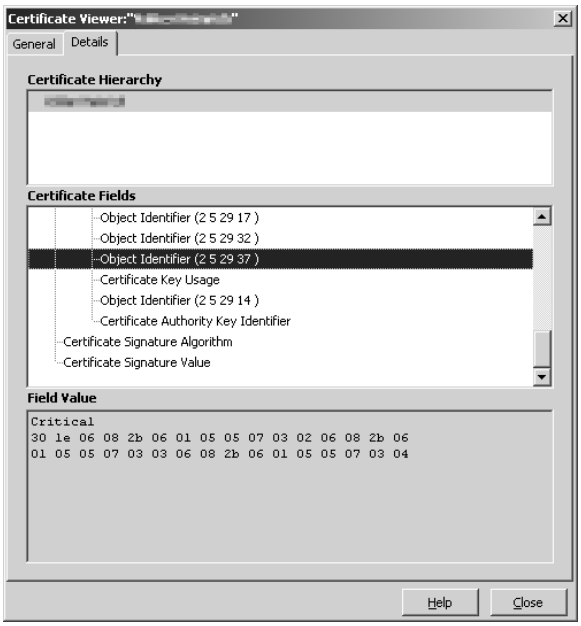

Figure 3.1: Critical *Extended Key Usage* extension, Mozilla certificate view.

vate extensions and encode the corresponding data into its certificates. An extension is simply a triple, consisting of a name-value pair of a globally unique object identifier (OID) and the corresponding value plus an additional flag that allows to mark the respective extension as "critical". If this flag is set, applications that do not know how to treat a certificate with a private extension (as they do not recognize the respective OID) must reject the certificate as invalid. For instance Mozilla in its certificate view does not resolve the name of some standard extensions properly, these extensions are only listed by their OID and the data is not interpreted (see Figure 3.1). Applications are free to ignore the additional information if the extension is non-critical. This rule easily renders the semantics of certificate contents unclear and could affect the interoperability of certificates negatively [114].

Even severe security holes can arise from interpretation elbowroom concerning extension attributes as experience has shown [230, 63]. Microsoft's CryptoAPI and the Linux web browser Konqueror were found to not handle the `basicConstraints` extension properly. This well-known extension indicates whether the corresponding key pair belongs to a CA or not (remember that the main characteristic of the centralized trust model is the fact that only CAs are allowed to issue certificates). CryptoAPI and Konqueror re-

garded certificates without this extension as CA certificates – in violation of the X.509 standard.[5] The consequences are indeed very security-critical with respect to, for instance, phishing web sites that can obtain a "valid" SSL certificate exploiting this weakness: A free email certificate from a CA which is a pre-installed trust anchor would do to issue a certificate for a web site of the attacker's choice.

Revocation Checking The last component of certificate validation involves CRLs or OCSP. Every certificate in the chain should be checked against a fresh CRL or with the help of an OCSP responder. However, these features are often implemented rudimentarily in current PKI-enabled applications [44]. Clients that are ready for CRL processing or OCSP access differ in the way they handle status information. For instance, Mozilla supports manual CRL import via HTTP and automatic updates under the given URL. Thunderbird (version 1.0) in turn does without status information at all.

The question of how to exactly check the revocation status of a certificate is often left as an exercise to the end user. The decision whether to use CRLs and OCSP at all and the choice what to do in case a fresh CRL or OCSP response is not available, is left to each individual application or the user respectively. As a consequence, the semantics may vary. Mozilla for instance regards a certificate as valid when OCSP is deactivated, but considers it invalid if OCSP is activated and the responder cannot be reached due to network problem. In contrast to that, Outlook Express (version 6) may tell the user that the certificate is "not revoked or the corresponding information could not be determined" indicating that it equates the status "not revoked" with the status "unknown". A uniform policy (including caching rules and the definition of update intervals) is thus hard to enforce with current standard clients. A related problem is the fact that certificates often do not indicate how and where the corresponding status information can be found (see below).

3.2.2 Managing Trust Anchors

We dedicate a whole section to the issue of trust anchors and especially certificate fingerprint verification for two reasons. On the one hand, the

[5] This bug is now fixed by Windows Service Pack 2 and newer versions of Konqueror.

(manual and/or out-of-band) verification of trust anchors is extremely important from a security perspective. On the other hand, unfortunately, there are very few mechanisms that make this process comfortable for users. Often enough this process requires the manual comparison of, say 20 hexadecimal characters.[6] As a consequence, users skip this step (implicitly or explicitly) and do not check trust anchors at all [92]. From a security perspective, this is intolerable.

PGP It is very likely that the awareness and experience with respect to manual fingerprint verification is highest in the PGP user community. The verification of new peers' keys is crucial for the functioning of the PGP web of trust as otherwise users cannot find appropriate certificate paths. Fingerprint verification has less relevance in the context of a centralized PKI, where it only occurs occasionally. End users typically ensure the authenticity of CAs' keys, which only account for a small portion of all keys. Nevertheless, the idea of openness forces end users sooner or later into adding new trust anchors as they communicate with men or machines outside their own PKI. There is a certain difference as a personal contact between the peers is more likely in the PGP world as with CAs. Commercial CAs offer hotlines where (in theory) the fingerprint of their certificates can be asked for over the telephone. There is some anecdotal evidence with the German division of VeriSign Inc. that this mechanism fails in practice as service employees do not know what to do in such a situation [154].

Windows Certificate Store Adding a new certificate into the Windows certificate store takes six steps, a process which has proven to make high demands on users [17]. Concentrating only on the fifth and most critical dialogue, there is criticism both from a usability and a security perspective. The dialogue, which is shown in Figure 3.2, at least violates the design principles of error tolerance, suitability for learning, and self-descriptiveness: It is difficult to undo a click on the "yes" button as this would mean to navigate through the certificate view in Microsoft's Management Console, a tool unfamiliar to most users. Furthermore, the window does neither emphasize how security-critical the task is, nor does it provide clues how to

[6] This figure holds for a SHA-1 or RIPEMD-160 fingerprint. As the security of the SHA-1 hash function has recently been called into question, longer fingerprints with 224, 256, or 512 bit (i.e. 28, 32, or 64 characters) will be used in the medium term.

verify the fingerprints. Besides, the translation of "root" into the German word "Stamm" (instead of "Wurzel") is non-standard terminology.

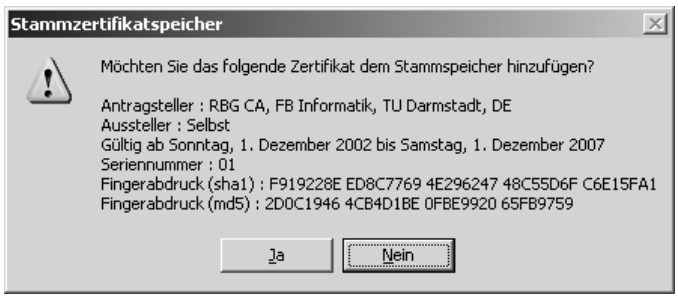

Figure 3.2: Adding a trust anchor to the Windows certificate store.

What is Missing As we can see, the system does not ensure that the user indeed verifies the fingerprint (which admittedly is a hard question) or at least is aware of her decision. Plus, the Windows certificate store does not keep a record, which would allow to later distinguish certificates that are pre-configured from trust anchors added by the user herself including for instance the date of import or the origin of the certificate. Compare this to the way software is downloaded and installed from the Internet – a process which can also have far-reaching security implications: A lot of people run anti-virus software and personal firewalls on their machine, which examine the code and stem malicious behaviour. Plus, software from unknown sources can be run in a quarantine environment, a "sandbox", where it cannot do harm. A similar mechanism for the "installation" of new root certificates does not exist. In our opinion this could be a promising starting point for further research.

The common way to shield end users from fingerprint verification (on the individual level) is to deliver pre-configured trust anchors in operating systems, browsers, email clients, and other PKI-enabled applications. For instance, more than 200 root CA certificates come with Windows XP.[7] It remains unclear which were the criteria the selection of the software manufacturer is based on and what security level is achieved (see also the discussion about policies in Section 3.2.3). It is therefore essential that users are

[7]Gutmann [116] extended this idea somewhat in putting third parties like ISPs in charge of providing clients with trust anchors. This process should happen transparently during dial-in, comparable to DHCP (dynamic host configuration protocol) bootstrapping.

– at least in principle – able to check the authenticity of keys on their own. Interestingly enough, some applications (e.g. Konqueror or Opera) do not at all offer the possibility to view certificate details including the fingerprint of CA certificates used to issue SSL certificates for web servers.

On the enterprise level, PKIs may decide to cross-certify, i.e. the root CAs mutually certify each other, or to join a bridge CA. A bridge CA is often a separate CA that issues certificates to existing CAs, making them a subordinate CA, or cross-certifies with each of them. The European Bridge-CA[8] for example interconnects PKIs from a handful of larger companies (Allianz Group, Deutsche Bank, Siemens, Deutsche Telekom) and some German trustcenters. It is no bridge CA in the strict sense, but uses a so-called certificate trust list, i.e. a container of root CA certificates which is digitally signed. The rationale behind this is to provide relying parties with authentic CA certificates, but to let them chose which ones to trust. Both the list and the signer's X.509 certificate can be downloaded by everybody from the web page. The PKCS#7 data structure however cannot be processed without additional software on a Windows platform therefore limiting its value for the masses.

Manual Fingerprint Verification Let us look once again at how PGP users handle the task of fingerprint verification. Fingerprints are distributed by various means, for example, they can be found on business cards, web pages, or email footers. Relying parties could compare the received fingerprint and the one computed out of the downloaded certificate to ensure the authenticity of the public key. So-called key signing parties, where people meet in person to mutually exchange their keys and mutually issue certificates, are popular in the community. Fingerprints can be compared over the phone by exchanging strings of hexadecimal characters or a representation which assigns words of the English language to each byte (e.g. the value F8 is mapped to the word "Vulcan").

Manual fingerprint verification has proven to be feasible for occasional use, but it does not scale well and can get tedious for a larger number of communicants. Therefore, some applications, for instance Secos[9] try to make things easier for users. Their idea is to present some sort of "cloze

[8] `http://www.bridge-ca.org`

[9] See `http://www.secos.org`. Secos is an application suite for secure email as well as secure instant messaging and file transfer. Its certificate and messaging formats are

test" that has to be filled in (see Figure 3.3.). This reduces the number of characters that has to be typed in manually, and on the other hand it ensures that the user really has gone through the verification process (as "guessing" the missing 8 characters by filling in arbitrary data is nearly impossible). We are in favour of such an approach as it reduces the overhead for users, allows to keep track of the keys that have been verified, and – where applicable – enforces the verification policy.[10]

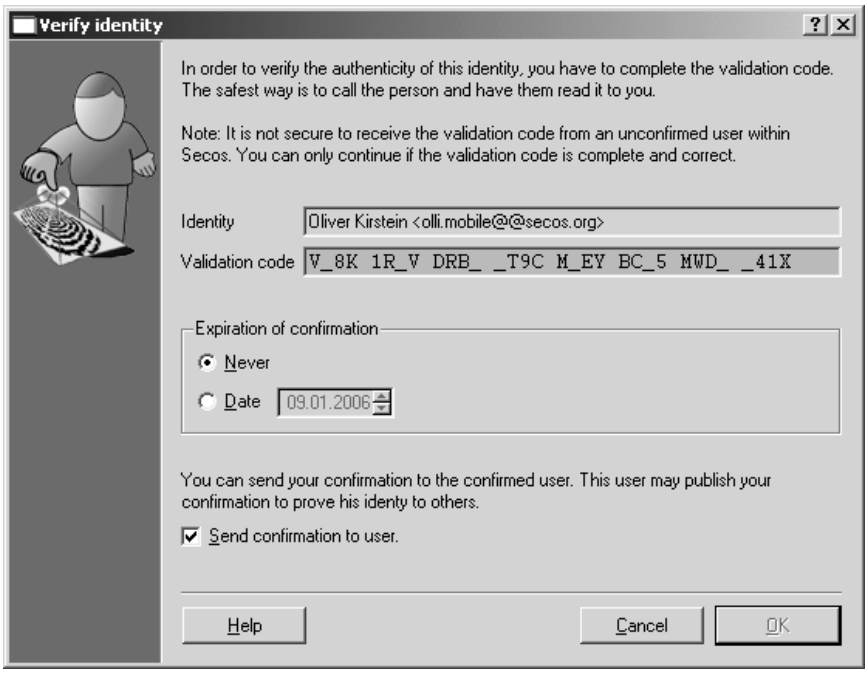

Figure 3.3: "Cloze test" for manual key fingerprint verification in Secos.

3.2.3 Certificate Policies

We have already touched upon policy issue in the two preceding sections. There is another area where security policies are relevant, namely a trust-center's certificate policy (CP), "a named set of rules that indicates the

proprietary, but the trust model has strong similarities to that of PGP, although the terminology is also proprietary.

[10] Observe that such a "cloze test" mechanism has to be designed carefully as the effective length of the hash value is diminished, which in turn increases an attacker's chance to find a key pair with the same fingerprint (second pre-image). The source code of Secos is not (yet) available, therefore the details of the verification method remain unknown (e.g. the hash function or the way the gaps are arranged).

applicability of a certificate to a particular community and/or class of applications with common security requirements" [52]. The purpose of a certificate practice statement (CPS) is to refine the CP and to document the practices which a trustcenter employs in issuing certificates. We can see in Chapter 5 that there are numerous security aspects a trustcenter has to take care of.

A relying party can judge the trustworthiness of a CA according to internal processes and protection measures as well as legal assertions (e.g. warranties, liability in the case of fraud) which are put down in writing in a CP and a CPS. In this section, we argue that the idea of having different security levels from CAs expressed in the corresponding CP and CPS is somewhat reduced to absurdity as there is no client-side integration or support to help users access and interpret this information.

Certificate Classes Important criteria are the type of the subject's PSE and the way how the identity of an end entity (a person or a machine, e.g. a web server) to be certified is checked: For instance, a very weak form of identity check is applied for gratis or low-cost email certificates where only the subject's ability to access a certain email account is verified. Some trustcenters call these "class 1" certificates as they only provide a low confidence and liability level. In order to receive a class 3 or 4 certificate, subjects must use a hardtoken and establish a personal contact to a registration authority of the trustcenter where they present their ID card and possibly further paper credentials (the labelling with class A, B, and C is a similar scheme). Some trustcenters, the trust anchors of which are pre-configured in common browsers, offer SSL certificates for about $ 50 a year. In this case, the identity of the organization running the server is checked only via a contact email entry in the DNS (domain name system) record.

It is obvious that there can be huge differences between various trustcenters and certificate classes. Despite this fact, current PKI-enabled clients do not indicate the confidence level one might have in a certificate. In particular they do not distinguish between high-assurance and low-end certificates in a way such that end users can percept it easily. There is a technical standard within X.509 to let a certificate refer to the policy under which it was issued. Inspecting the certificates that come with a fresh Windows XP installation, one finds out that only few of the commercial trustcenters make use of the

corresponding `certificatePolicies` extension. We do not conceal the fact that including this extension anyway does not help very much:

Open Issues Firstly, it only contains the OID of the CP and/or CPS or a hyperlink to these documents at best. According to Gutmann [114], earlier VeriSign certificates used to hard-code parts of the policy directly in the certificate or even in the Outlook Express client (sic!). A more recent example are certificates of the German root CA for qualified digital signatures. These certificates in fact include a policy identifier (1.3.36.8.1.1), but one cannot obtain the policy from the Bundesnetzagentur (as the operator of the trustcenter) neither via WWW, nor by email.

Secondly, the interpretation of the CP is left to the user alone, there is no easy and user-friendly way to assess the trustworthiness and confidence or liability level at a glance or to compare different policies. Some CAs have gone into the habit including "class x" into the DN, but this would only help if users inspect the certificates manually. However, if they do so, they should be prepared to sometimes discover strange name components like "OU = NO LIABILITY ACCEPTED, (c)97 VeriSign, Inc.". This deficit also complicates the management of trust anchors for users. Consider for instance the new PKI of the Fraunhofer-Gesellschaft that plans to issue certificates both to employees (and servers) of the Fraunhofer-Gesellschaft as well as for external business partners [20]. While internal users will be equipped with hardware PSEs, external users will only receive softtokens issued in a simplified enrolment process. Consequently, two different CAs for internal and external users respectively will be set up. However, those two CAs could not be joint together under a common root CA (which would make validation easier). A single trust anchor would not allow to distinguish between the different policies and security levels. One thus had to resort to use two disconnected root CAs and to hope that relying parties will recognize the difference when importing the trust anchors.

3.2.4 Information and Service Access

Fundamental tasks like retrieving a communicant's certificate or obtaining a key pair and a certificate pose major hurdles to PKI users as it is often unclear how and where to find the necessary information.

Getting One's Own Certificate Obtaining a key pair and the corresponding X.509 certificate is harder as it sounds. In contrast to the PGP community. where user can easily generate their own keys "on the fly" (with the PGP software itself) and distribute them in in the form of self-signed certificates, this is no common practice for X.509-based clients in the light of the centralized trust model. PKI-enabled applications offer some support for end users, but the integration with the certificate application process is very loose. Outlook's dialogue windows for the configuration of user key pairs and certificates for digital signature and decryption contains a link to Microsoft's web pages. Here three certification service providers from the U.S. are listed that offer class 1 certificates for free or for $ 20 respectively. Outlook Express also contains a button "request digital ID", but nobody at Microsoft seems to care that the corresponding hyperlink is broken. Mozilla refers to a web page with more than a dozen, international CAs.

In most cases key pairs can be generated and the corresponding certificates request through a standard web browser, which is favourable from a usability point of view. However, experiences indicate that it "takes a skilled technical user between 30 minutes and 4 hours work to obtain a certificate from a public CA that performs little to no verification" [116] – needless to say that this is deterrent. There is a particular difficulty when using non-Microsoft email clients and/or alternative browsers on a Windows operating system: users may be forced to find out how to export a key pair from the browser, which generated it and transmitted the request to the CA, and import it into the email client.

Getting Other People's Certificate Let us now turn to the problem of retrieving a communicant's certificate. The most common mechanism to store certificates and make them available to relying parties in an X.509-based PKI is a central LDAP (Lightweight Directory Access Protocol) directory maintained by the trustcenter [8]. Although LDAP is a relic from the time of X.500 [51] directories (that "invented" X.509 certificates as their authentication mechanism), it is supported by current email clients. The lookup of a certificate works fairly well (and merely transparent with the sole exception of the short delay an LDAP request may cause) provided the email address of the communicant and the network address of the correct LDAP server is known (and there are no multiple entries per per-

son [115]). The latter condition can be a problem as companies do not open their directory to the general public at all or do not publish its address (in order to hide possibly sensitive information, e.g. their internal structure). Especially there is no "canonical name" for the certificate directory belonging to a certain domain name. Compare this to the situation with the World Wide Web. If one knows the email address, for instance `tstraub@cdc.informatik.tu-darmstadt.de`, of a person, one can easily infer the name `www.cdc.informatik.tu-darmstadt.de` of the web server of the corresponding organization by adding the canonical prefix to the domain part. PKI would benefit from a similar established mechanism for certificate lookup, e.g. by a canonical name or an extension to DNS. (Needless to say that the problem of certificate lookup goes hand in hand with the difficulty of finding out the configuration settings for e.g. OCSP responders or CRL distribution points when they are not explicitly indicated in the certificates themselves.)

What Can Be Done In an X.509-based PKI, a common, yet ad-hoc solution to disseminate certificates for encryption keys is to sign all outgoing mail automatically. RSA key pairs can be used both for encryption and digital signatures, so a recipient of a digitally signed email may answer the sender in encrypted form. This requires not only that a common key pair is used, which may interfere with the need for key backup or shared use of credentials (see below), but that the recipient can successfully validate the sender's certificate. If the corresponding trust anchor is unknown to the recipient, current implementations create more cognitive and operational overhead showing warning messages and demanding immediate security decisions from the user [92]. Therefore this approach is only of limited value.

The problem of certificate dissemination is elegantly solved with PGP: There are only a handful of so-called key servers, they synchronize with each other, one or more key servers are already pre-configured in PGP-enabled applications, and new public keys can be published easily (even by the key holders themselves or third parties). Secos implements a similar mechanism in that it uses only a single, system-wide server. In this respect however, it is a closed system violating our postulate of openness. For instance, for secure messaging users have to set up a new email address on the Secos server (those addresses are internally indicated by a double @ sign, see Figure 3.3).

Clearly, a concentration on only a few certificate directories will not happen with centralized PKIs, not least because organizations or trustcenters may not want to loose control over "their" data. An interesting alternative, however, is the setup of intermediaries in the form of meta-directories that offer a "single point of entry" to a multitude of directories. This idea is realized by the so-called LDAP proxy described in [76]. Such a proxy acts as a mediator, transparently passing on client requests to a possibly large number of physical directory servers. Among the advantages of the proxy are its centralized configuration and the possibility to filter out sensitive entries before returning an answer to the client. We are in favour of such approaches as they facilitate the pure technical and auxiliary tasks without affecting security.

In order to promote the dissemination of directory information, we suggest a new, heuristic method how email users could extend their list of LDAP directories. Assume that an incoming email carries a digital signature. Current clients also include at least the sender's certificate or even the whole certificate chain in order to assist the recipient in signature validation. As clients have to parse the certificates anyway, they could automatically add to their local list pointers to directories extracted from, for instance, the `CRLDistributionPoints` extension if present. Again this approach does not affect security, as new certificates are always validated prior to use. But if the recipient already knows the authentic trust anchor of one communicant of the PKI, she may benefit from this heuristic as she can then use other certificates from that PKI, too.

3.2.5 PSE Management

Last but not least we point out some challenges in the area of PSE management. It is a characteristic of PKI that, with end-to-end security, responsibilities get shifted towards the end users (cf. page 56). In this context challenges can arise from the requirement to handle passwords (which are used to secure private keys), to manage key rollover and the renewal of certificates, as well as practical business demands like vacation replacements. Here we do not treat passwords further as they have been discussed in Chapter 2 and they are also the subject of one of the sample modules in Chapter 4.

Key Rollover Key rollover is a periodic process in a PKI. It means that an entity changes its key pair and obtains a new certificate. This process is also called key/certificate renewal.[11] We have seen in Section 3.2.1 that timely key renewal is an issue with SSL server certificates. Let us now look on the client side: PKI-enabled applications that use PSEs allow no automation of rolling over to a new key. This again has negative implications for usability, as there is no built-in mechanism. While this is typically not so critical for keys that are used for authentication or digital signatures, it may cause trouble with encryption keys, e.g. when used for secure email. Encrypted emails are stored in the local mailbox in the same form as they were received by the MUA (holds both for S/MIME and PGP). Subsequently the user has therefore to provide her PSE each time when she wants to read such messages again. In course of time, she changes her key pair over and over, but she also has to keep a key history of old decryption keys the certificates of which expired long ago. This is not only problematic from a security angle (as old keys or algorithms may be easier to attack), but also from a usability perspective: a growing number of PSEs has to be kept securely and loss of one PSE implies data loss as well. Besides, the user is forced to work with multiple devices if her private key are stored on hardtokens. Current S/MIME-enabled MUA cannot cope with this problem. Additional software like *FlexiTrust 3.0 ReCrypt Tool for Outlook* [82] has to fill the gap as the MUA does not allow to decrypt all mails encrypted with an old key and write them back on the hard disk re-encrypted with the current public key. However the author knows of no such tools for other clients like Mozilla or Thunderbird. Related to key rollover is the process of key backup, i.e. the storage of copies of private keys at a secure location where they can be regained from when the original key gets inaccessible. Prudent PKI design always takes this question into consideration (see [20] for an example).

Delegation of Private Keys A typical situation is where an employee is on holiday and the proxy should be given the possibility to access her email. Assume that Alice has key pairs she uses for her business, individual email and for authenticating at web portals. There are standard techniques

[11] One typically assumes implicitly that not only the certificate changes, but also the key pair. CAs may issue a new certificate for the same key – this is called "certificate update". From a security point of view it is desirable to always generate a new key pair, so this is no common practice.

to forward email temporary or permanently to another address which Alice can use if she is on holiday or a business trip and wants her replacement Bob to access her mailbox. However, things work only fine as long as emails are unencrypted as otherwise what gets forwarded to the proxy are unintelligible messages. In Section 5.5.1 we show how one can solve this problem without handing out the private key, neither to the mail server nor to the other person.

In Chapter 7 we present a solution for *delegating* credentials in the form of a private key for authentication. Temporarily delegating the ability to access a service or resource on the web to a colleague is a natural requirement, but it is rarely supported originally on the server side. Consequently, people have no choice other than sharing the respective PSE. Obviously, this has negative security implications, especially when the same secret is used for multiple purposes, e.g. the key pair is used for authentication and encryption. Besides, this naive method does not allow to restrict the proxy's capabilities to a certain period of time or prevent him from further distributing the secret.

3.3 PKI Evaluation Tool

In this section, we present our tool to measure the usability and utility of PKI-enabled applications with respect to their security features. Our evaluation tool can also serve as a design guideline and systematic check list for software developers who have to cope with the before-mentioned challenges. Sample applications of the tool are the topic of Section 3.4 and 3.5. The evaluation and rating process resembles that of evaluating security products as proposed in the Common Criteria [1]. Since the Common Criteria methodology is an ISO standard and receives worldwide acceptance, we consider it a reasonable basis for our approach, too. Throughout our framework we often use the term *user*. Depending on the concrete scenario, this may either denote an end-user or an administrator.

Before the evaluation categories are described in detail in Section 3.3.2 to 3.3.4, their organization and the idea of usability profiles are explained in the first two sections.

3.3.1 Organisation

3.3.1.1 Evaluation Categories

In all, our framework consists of 15 evaluation categories which are evenly spread in three groups named *deployment issues*, *ergonomics*, and *security features*. The motivation behind this choice is Schneier's famous saying "security is a process, not a product" [206]. We thus have to follow an integral approach and must not restrict ourselves solely to ergonomics or technical features. To illustrate the ideas behind each category and to simplify the evaluator's task, we propose a number of exemplary questions. These *central questions* have to be investigated throughout the evaluation to obtain a category score.

In order to make use of a PKI-enabled product, it has to be deployed. For this reason, the first category group is named *deployment issues*. For instance, the user has to obtain the product and install its components, which often means that he has to be familiarized with the PKI features before being able to use them. We turn in detail to the deployment issues in Section 3.3.2. With the unmotivated user property P1 of Figure 2.4 in mind, this category should not be underestimated.

Secondly, the handling of a PKI-enabled application in service is discussed in the category group *ergonomics*. In a classical sense this category group is the core of any usability. Typical categories of ergonomics are the composition of the menus and dialogues, respectively, and the help system available. We present the central questions of ergonomics in Section 3.3.3.

The categories defined so far may easily be adapted to other use cases than PKI. However, the best product with respect to ergonomics is pointless, if the use case specific security features are substandard. Our third category group *security features* takes the properties P4 (barn door) and P5 (weakest link) into account. It deals with the algorithms and key lengths in use, the certificate management, and how the validation of a certificate may be done. We turn to the security features in Section 3.3.4.

3.3.1.2 Usability Profiles

In order to obtain a quantitative measure, an evaluator will award points for each category. The result of each category is an integer score in the interval $[-2, 2]$ where 0 denotes an average result, ± 1 a little above/below average

result and ±2 an outstanding/substandard result. The guidelines on how to assign points are part of a so-called *usability profile*. A usability profile has to be defined before the evaluation actually takes place. This can be done based on the central questions of our framework. Each usability profile has to list sample answers and a guideline on how to award points. We provide appropriate clues in the following sections.

The definition of the usability profile may, for instance, be done by the evaluator himself or by a usability specialist involved in the process. An evaluation result has always to be interpreted with respect to the underlying usability profile. We point out that this procedure is analogous to the definition of so-called protection profiles as described in the Common Criteria.

Our framework allows for an adaptation to different environments since the partial results of a category group are weighted to get an overall score. The weight definition depends on the concrete usage scenario and is part of the usability profile. Figure 3.4 shows three coarse examples for scenarios with *high security* requirements and skilled personnel (e.g. in the military or a research department), *enterprise* usage with low deployment and helpdesk budgets (with average security demands), and a *private* usage scenario. Let us quickly explain the proposed weighting in Figure 3.4 in case of private usage: A private user is in general not supported by an administrator, that is he has to deploy the PKI-enabled product himself. In addition, the unmotivated user property P1 requires an easy to deploy and easy to install application. Therefore, deployment is rather important that is why we assign a weighting of 40% to this category group. The properties P2 (abstraction), P3 (lack of feedback), and P4 (barn door) motivate a weighting of 40% for the group ergonomics, too. Due to the small assets and threats in this sce-

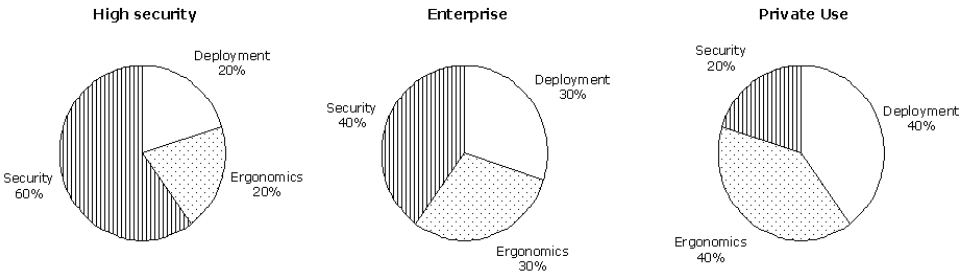

Figure 3.4: Sample weightings of the category groups.

nario, the security features are of secondary importance and thus weighted by 20%.

Table 3.1 gives an overview of the category groups. The individual categories are explained in the remainder of Section 3.3.

Deployment Issues
Gathering Information and Obtaining the Software
Technical Requirements
Installation and Configuration
Technical Support
Training

Ergonomics
Menus and Dialogues
Transparency of the Security Features
Warning and Error Messages
Help System
Reaction in Potentially Threatening Situations

Security Features
Algorithms and Parameters
Secret Key Handling
Certificate Management
Status Information
Advanced Functionality

Table 3.1: PKI evaluation categories.

3.3.1.3 A Note on Evaluation Methods

To give the evaluators a maximum degree of freedom, we formulated the framework in a way that does not prescribe a certain usability evaluation method. This is mostly of interest for the category groups of deployment and ergonomics, but may also have a certain relevance for the security features. Consider for instance the way of how end users manage certificates and trust settings.

Two classes of usability evaluation techniques are prevalent in the HCI area today, namely empirical and informal methods; automatic and formal methods only play a secondary part [176, 174]. Empirical methods assess us-

ability with the help of "real" users, i.e. test subjects that are representative of the actual user circle. In contrast to that, informal methods rely on the ability of expert evaluators that inspect the user interface themselves. We applied both types of methods for the evaluation of usability-related issues in the context of our framework. A heuristic evaluation can be performed based on the central questions given for each category in the generic version of the framework or a specialization thereof. A cognitive walkthrough method can, for instance, be used to pass through the steps necessary for the initial system setup. For the Microsoft Usability Study we used a combination of both approaches in a "pluralistic" way (as not only a single, but two or three evaluators at a time inspected the interface together). For the evaluation of trustcenter software we added a laboratory test based on plausible usage scenarios.

3.3.2 Deployment Issues

First we turn to the category group *deployment issues* which describes the prearrangements necessary to set up the system. The central questions listed in the following mainly address property P1, i.e. the user's little motivation to really use built-in security features of existing software or to switch to another application of the same kind for the sake of better support of PKI features. Suppose that a computer administrator or an end-user chooses to increase the level of enterprise-wide or personal security, respectively. Then it should be made as easy as possible to realize this plan. People who once tried to improve security but did not succeed are likely to get frustrated and to give up.

In all, the deployment issues comprise the following categories: The first step is *gathering information* about candidate products which is closely related to the question of how the software itself can be *obtained*. Since sophisticated applications tend to have special infrastructure demands, it is necessary to consider *technical requirements* before *installing and configuring* the application. During the latter two steps, there may arise the need for technical *support*. Finally, it should be assessed how much *training* is required for the product to be used securely. We present in detail the central questions for each category in what follows.

3.3.2.1 Gathering Information and Obtaining the Software

If an evaluator has no a priori knowledge of appropriate products, it is important that comprehensive information, maybe even an evaluation version, of the software is easily available. In addition, analysis from professional journals or experience reports found in news groups provide more neutral and objective information. It goes without saying that the easier information and the product itself are available, the better the rating is in this category. Central questions are:

- How much meta information (surveys, tests etc.) comparing different products of the same kind is available?
- Is there already enough experience about a product of recent date?
- Does the vendor offer precise product information on his web pages?
- How complicated is it to obtain the product, can it be purchased online?
- Is it possible to upgrade an evaluation copy to a full version simply by an activation code, i.e. without re-installing and re-configuring the software?

3.3.2.2 Technical Requirements

A successful installation often depends on the technical environment in use. For instance, the hardware of a five-year old PC is not sufficient to run current server applications due to performance and memory restrictions. Again, we refer to the unmotivated user property and stress that a good PKI-enabled application should make it easy for users getting started with it. It should ideally not require high-end hardware nor additional software (e.g. special plugins) to do its task properly. These requirements can only be rated in the light of the actual prerequisites, since some software, e.g. for a certification authority, may in fact have very particular needs. We thus propose that the easier it is for the user to fulfil the requirements of the software, the better the application is rated. In this context, “easier” has to be interpreted in terms of time or money.

- Is it possible to install and operate the application with existing hard- and software?

- Are multiple operating systems supported? Which ones?
- What about standard compliance and the application's compatibility with auxiliary programs?
- Does the installation require additional programs?
- If the application needs additional hard- or software: Which/how many third-party products are supported?

3.3.2.3 Installation and Configuration

The background of this category is that proper installation and initial configuration are crucial to security with respect to the barn door property P4. On the one hand, there should be enough support, e.g. an installation wizard or a list of commonly used configurations, to guide the user. On the other hand, the default configuration must provide an appropriate security level. This is due to the consideration that it may be a security novice who is installing the software and who is above all not interested in the security features themselves but in getting the software running. An example is the selection of cryptographic parameters like key sizes. While this does not affect the functionality of the software, it is indeed security-relevant.

- Is there an installation wizard guiding an inexperienced user?
- Does the product offer to choose from predefined scenarios?
- How time-consuming, how difficult is it to adapt the configuration to individual needs?
- How does the software check user settings for consistency and security?
- Which level of security does the default configuration provide?
- How much expertise is necessary to successfully install and configure the application for the first time?

3.3.2.4 Technical Support

Even if the PKI-enabled software may have advanced installation and configuration wizards, a user may sooner or later need additional help. Therefore,

we direct our attention to the quality of support, too. Once again, the criterions in this category strongly depend on the actual application scenario. The rating scheme is analogous to that of the category *technical requirements*.

- What kind of support does the vendor (the supplier) offer? How expensive is that support? Is it charged per request or is there an annual fee?
- How can the user contact the support team? (e.g. phone, email) At what times? (7x24 or office hours only)
- Are there FAQ or manual pages on the Internet?
- Is the information provided in the user's native language?

3.3.2.5 Training

The deployment of every new application should ideally be accompanied by training lessons for users. As we all know, this is more wishful thinking than reality. Due to the abstraction property (P2), teaching users to deal with security-related tasks may in particular gain importance where ergonomic deficiencies have to be compensated.

- Which amount of training for the intended users is absolutely necessary to provide an appropriate usage?
- Does the software provide an interactive guided tour explaining its basic functionality?
- How many companies offer training lessons? At what price?
- Which books provide course materials?

3.3.3 Ergonomics

One may argue that *deployment issues*, which we have discussed in the previous section, also fall into the range of usability. To make clear the following difference, we decided to name the current section "ergonomics": Contrary to *deployment issues*, which cover non-recurring actions during the setup process, *software ergonomics* is an everyday concern which in

general affects much more tasks and people. It may be tolerable that the installation phase requires expert knowledge (which means that the category group *deployment issues* is assigned a smaller weight in the usability profile). But this does not hold for the "operational" phase where people are expected to actively use the PKI-enabled application to get their work securely done.

A natural category to start with is the UI design concerning *menus and dialogues*. The information given there should be self-descriptive using a consistent terminology, whereas its organization should try to anticipate how users will operate the application. Because of property P1 (unmotivated user), security software should follow the maxim to hide complexity from the average user, acting in a seamless way. Hence, the second category is the *transparency of security features*. Since it cannot be guaranteed a hundred percent that the respective PKI-enabled application will always work without interaction, *warning and error messages* are treated in the corresponding section. It is exactly in this situation when a user may question the *help system*, probably for the first time (à propos "trial and error", cf. Figure 2.3). From the viewpoint of security, it all depends on how the application *reacts in potentially threatening situations* since errors should be prevented at all costs.

3.3.3.1 Menus and Dialogues

This category groups two types of UI components, the menus being the more static and dialogues being the more dynamic elements. Once again, we think of the user as a security novice and unmotivated subject (P1). Therefore, on the one hand it should be possible to intuitively find *all* the security-relevant menu entries because of P5 (weakest link). On the other hand, dialogues should show a careful design due to the lack of feedback property. Where users are supposed to change security parameters, there should be clues to underline sensitive settings and a possibility to restore a previous or even a default configuration. A general requirement is that technical terms stemming from PKI and cryptography should be used in a precise and consistent manner.

- Does the organization and structure of PKI-relevant menus follow the general principles used throughout the application?

- How are the security-relevant menu items organized? Are they easy to find and grouped reasonably?
- Does the visual design of the dialogue reflect its level of security relevance? (e.g. special symbols for low, medium, high security relevance)
- Are the technical terms chosen well and intelligible? (even more important if the application does not come in its native language)

3.3.3.2 Transparency of the Security Features

This category refers to a common paradigm to encounter the unmotivated user property. UI experts agree that an ordinary user should at first get into contact with as few security features and settings as possible. However, the more he is interested in the details, the more information the application should provide. This serves a twofold purpose: First, to increase a user's trust in the application by giving further insights. A second aspect is the suitability for learning (cf. Figure 2.3). A standard technique, for instance, to both hide complexity and allow users to get deeper into it, is an "Advanced Settings" button.

- To what extent does the user have to get involved in security settings to be able to use the features?
- Do the default settings meet the requirements of an ordinary user?
- How difficult is it to change the transparency settings?
- Can a user adjust the transparency level according to his individual skills? (e.g. ordinary user, interested layperson, expert)
- Is every necessary piece of information accessible at all? (e.g. details about cryptographic algorithms and protocols)

3.3.3.3 Warning and Error Messages

In situations where the user or his communication partner behaves wrong or in a non-standard way, this category plays an important role. When faced with a warning or error message, a user's decision is often crucial for security. The user must understand the warnings and behave in an appropriate way. Otherwise he will just click the "close" or the "continue" button. It is

therefore indispensable that an explanation of alternative actions and their consequences is given. A good practice is not to force the user to make an immediate decision, which is the case when using modal dialogue windows, but to give him the possibility of proceeding in another way, e.g. browsing the help pages or postponing the task. Consequently, the abstraction property, the barn door property, and the weakest link property are the basis for the following central questions.

- How many details are given in message dialogues?
- Does the text explain what to improve if a single step of a complex PKI operation, e.g. signature verification, fails?
- Are there warnings in case of a security-sensitive operation? What advice is given in such a situation? (e.g. if the chosen key lengths are too small)
- How precise, understandable, and insistent is the information given? Does the message encourage to ignore it?
- Are there useful links to matching topics in the help system?

3.3.3.4 Help System

The built-in help system of an application is the primary source to get information during usage. The help system should be designed to suffice an average user in most of the cases. We think it advisable to have a short glossary of commonly used technical terms of cryptography and PKI, as well as a short introduction to the topic on a rather easily intelligible level, but with links to more detailed information. A nice feature would for example be an integrated FAQ section. While it is common practice to refer the user to the vendor's web site for further help using hyperlinks in dialogues, we vote against it. For the sake of convenience this mechanism should only be used for non-critical, additional, and up-to-date information. But not for answering standard requests since a user may not be connected to the Internet at the moment a question arises. We consider it a must to have context-sensitive help in all security-related menus and dialogues, at least in the most critical ones. The central questions touch the same areas (P2, P4, P5) in the current category as in the previous category.

- Which of the main technical terms of PKI and cryptography are explained? (e.g. certificate, key pair, certification authority) Are the explanations correct and intelligible?
- How detailed is the introduction to PKI and its concepts? Is the text written in an abstract fashion or are there useful examples?
- Is a context-sensitive direct help implemented? If yes, in which way does it support the user? Are there further links to the help system?
- In which way is the user assisted when accomplishing a complex task? (e.g. when installing his own private key)
- Are there links to external information sources about PKI and current developments?

3.3.3.5 Reaction in Potentially Threatening Situations

In order to get a reasonable measure on the utility of a PKI-enabled application, we have to test its reaction in critical situations. We consider this worth a separate category due to the potentially high damage that may be caused in case of failure. A typical scenario is a digitally signed message which has been tampered with. Then the application has to reject the document and explain in detail not only the reason of failure, but give a concrete guidance which measures the user has to take next. In case the local key or certificate store is manipulated, the application must react accordingly and trigger an alert. It is also important how status information is handled. Suppose, the application wants to use an X.509 certificate, which specifies the URL of a CRL distribution point or an OCSP responder, but the corresponding server is unreachable.

- What is the behaviour of the application in case of a certificate that cannot be validated successfully due to an unknown issuer?
- Does the application tell the user how to check the fingerprint of a certificate when introducing a new CA to the system?
- How does the programme react in case a source of status information specified in the certificate is unreachable?
- Does the application has the ability to automatically check for and download security updates and patches?

3.3.4 Security Features

The subject of the third and last category group are the security features of the PKI-enabled application under evaluation. These features relate indirectly to the issue of usability as decisions on the technical layer often have an influence on UI design. If, for instance, the application does not implement functionality, which is necessary for using PKI in a certain environment or does not support a common standard, users are urged to find a way to work around this problem (probably weakening the system). Security on the cryptographic layer and utility of a PKI-enabled product are thus crucial factors for user acceptance, too. With regard to the barn door and weakest link property (P4, P5), we emphasize that the application should provide a proper implementation of the security mechanisms since it is virtually impossible for the user to detect flaws on this layer.

The arrangement of the following categories is relatively straight-forward: First of all, we review *algorithms and parameters* which belong to the cryptographic layer. The next category revolves around the *handling of secret keys*. This leads us to the *certificate management* and the question of how the application deals with *status information*. A last category is named *advanced functionality*.

3.3.4.1 Algorithms and Parameters

In order to achieve compatibility towards different communication partners, an appropriate set of cryptographic algorithms should be available. We emphasize that a reasonable selection of methods should include standard and widely deployed algorithms. This does not call for as many as possible and moreover exotic algorithms as this may bear additional risks (at least to confuse the user). However, it is important that the cryptographic building blocks rely on independent mathematical problems, e.g. integer factorisation and discrete logarithm in an elliptic curve (EC) group. This is because of the observation that mathematical or technical progress may render a single algorithm insecure. The cryptography inside the application cannot be treated separately since its effectiveness also depends on the protocols built on top of it. Thus, the word algorithms extends to protocols in this section. The second aspect in this context are the cryptographic parameters, typically including key sizes and lengths of message digests.

- How many different combinations of cryptographic algorithms and parameters are available?
- Does the selection comprise state-of-the-art algorithms? (e.g. AES, RIPE-MD, EC cryptography)
- Is it possible to choose individually from the set of combinations of algorithms and parameters? Does the system warn against weak combinations?
- Are all weak combinations disabled in the default settings?
- Are cryptographic primitives based on independent mathematical problems?
- Does the application implement a strong pseudo-random number generator? What source of random is used as seed?

3.3.4.2 Secret Key Handling

This category is about how secret keys are generated, protected, applied, transferred, and destroyed by the application. In this paragraph we use the term secret key in a generic way. While it is often replaced by the term private key in the context of a public key scheme, a secret key in a narrower sense is used for a symmetric algorithm, e.g. as an ephemeral session key. It goes without saying, that secret keys have to be protected carefully by versatile technical precautions.

- Is the application able to generate keys and key pairs on behalf of the user?
- What kind of storage devices for secret keys are supported? (e.g. SmartCard with corresponding reader, USB token, hardware security module, or softtoken)
- Which cryptographic and other technical precautions are provided to secure a secret key when stored on the hard disc or in memory?
- How is a secret key enabled for usage? Is it possible to enable a batch processing? (e.g. for bulk signature generation)

- In case a secret key is exportable: Are there efforts to enforce a certain password complexity when using password-based encryption?
- Can key data be securely destroyed, e.g. by multiply over-writing the corresponding area on the hard disc?

3.3.4.3 Certificate Management

We now turn to the management of third party certificates. Two things are important in this context: In our opinion, the application should first support the import of certificates via different channels to facilitate this task for the user. This means that a lookup on a certificate server or – in the PGP terminology – *key server* should be supported, using different protocols like LDAP, HTTP, FTP, or simply retrieving the certificate from the file system. The second question is how the integrity of the certificate database is guaranteed. If the certificates are stored on the hard disc, a typical method is to use a password-based MAC or a central storage location to ensure that the database has not been tampered with.

- How does the import and retrieval of certificates take place?
- Is it possible to read/write a certificate from/to a hard token?
- Does the application cope with certificate trust lists (CTL) for import or export?
- Which technical measures are taken to protect the certificate store?
- How can the user assign and withdraw trust to third party certificates? Is this possible on a fine-grained basis?
- Can the certificate handling be delegated to a central point or authority? (e.g. the OS or an administrator)

3.3.4.4 Status Information

Especially with regard to signature verification in the context of non-repudiation, the ability to gather information about the validity of a certificate at a certain point of time is crucial. It is also important to guarantee a timely revocation of public keys used for encryption. However, status information often is not retrieved automatically or processed properly by PKI-enabled

applications. Today, different methods are available to answer a validation request. We point out that the answer to such a request is nontrivial, as there are different validity models.

- Does the PKI-enabled application automatically gather status information?
- Which mechanisms for status information are supported? (e.g. CRL, delta CRL, indirect CRL, online requests via OCSP, see [52])
- Does the application make use of a CRL distribution point entry in an X.509 certificate?
- Does the application allow to manually import status information obtained from another source?
- Is there a mechanism to archive validation material?
- Does the application ensure that the machine's local time is correct?

3.3.4.5 Advanced Functionality

Finally, we come up with some additional features. While some of these features are well known in the community, they are often not implemented by current PKI-enabled applications. Particular use cases, like for instance the processing of signed documents which is subject to digital signature legislation, may have very special requirements. In environments with high security demands, e.g. for governmental use, a certain level of security evaluation may be an absolute must.

- May an administrator enforce a group policy concerning e.g. key export or maximal usage periods for cryptographic keys?
- Is the application ready for enhanced concepts like cross-certification or bridge CAs? Are time stamping services supported?
- In case the application processes X.509 certificates: Which extensions are implemented, are critical extensions always handled correctly? Does the application require proprietary extensions? Is the application compliant to major standards? (e.g. the PKIX or ISIS-MTT profile)
- Does the application support key sharing/backup? (e.g. à la Shamir [211])

- Has the application undergone an evaluation of its security features? (e.g. according to the Common Criteria)

3.4 Microsoft Usability Study

The experiences made in the course of the usability study carried out on behalf of Microsoft, stimulated the development of the formal PKI evaluation tool which is the core of this chapter. In order to give an insight in the development process and practical aspects of the evaluation, we give a short overview of the study in the following. Due to space restrictions, only a small fraction of the information can be presented here. The interested reader is strongly advised to consider the original work [44].

3.4.1 Objectives and Scope

To cut a long story short, the focus of the study can be stated as follows:

- to motivate the need for PKI-enabled applications with respect to current security threats and requirements,
- to illustrate the various use cases that have already been supported (in large parts even by standard and/or free software),
- to show shortcomings and common pitfalls applications suffer from,
- and to test and compare products for selected use cases according to a uniform evaluation methodology.

The application scenarios covered by the study include the protection of communication channels on the Internet (via SSL), secure email (according to the S/MIME standard), the security of groupware, file encryption and digital signing of documents, access control and token-based logon, virtual private networking, and the protection of wireless networks. In addition, cryptographic application programming interfaces (APIs) are also reviewed. In total, the study covers a dozen different use cases and three products on average for each use case. The names of the products can be found in Table 3.2 (page 102). We originally planned to investigate PKI-based services in SAP like secure workflows, but we did not find evaluators with sufficient previous knowledge in this area.

3.4.2 Methodology

Structure For each use case, we conducted a market analysis and chose the most common (according to third-party surveys and usage statistics) products for our evaluation. The respective software makers were then asked for an evaluation version, which was successful most of the time. To give the reader of the study an impression of the security features and the operation of the application, we described at least one of the products in detail, providing technical background information and screenshots. This part was subdivided into sections covering installation, ergonomics, and security features. These systematics match the category groups the products were actually evaluated according to. The results for each product were presented in a table providing marks and a textual explanation.

As the categories used in the Microsoft study are quite similar to those used in our framework, which is the subject of Section 3.3, we refrain from enumerating them here. The differences between the study and the framework consist of generalisations ("status information" instead of "revocation lists", "secret key handling" instead of "management of one's own key pair"), logical combinations ("installation and configuration", "menus and dialogues"), and additions of categories ("gathering information and obtaining the software", "reaction in potentially threatening situations").

Test Subjects The test subjects for the product tests were two faculty persons having a "Diplom" degree in computer science as well as 16 main course university students (having already earned a "Vordiplom" degree in computer science or mathematics). Most of the students (14 of 16) came from Technische Universität Darmstadt. They were recruited out of the lecture "Introduction to Cryptography" given by Prof. J. Buchmann in the winter term 2002/2003 (4 hours lectures plus 2 hours exercises a week). The idea and concept of the study had been presented during the lecture. At the end of the winter term, the students had to undergo a written exam the results of which we took as a criterion to chose potential test subjects. We send an email to the 38 of the 230 overall exam participants whose mark was 1.0 or 1.3 (scale ranging from 1 to 5, average result: 2.59). In return for participating in the study, students were offered to choose from either earning some money (60 hours short time contract, approximately 500 EUR) or credit points (equivalent to a seminar).

Evaluation Procedure Before the evaluation started, the authors of the study briefed all the test subjects about the objectives and explained the test framework to them. Test subjects were supervised and assisted during the whole study to ensure a standardized evaluation procedure and the comparability of the results. The test subjects worked together in groups of two or three on one of the use cases listed in Table 3.2. Some of the use cases consisted of a server and a client component like Groupware or VPN. Here the same test subjects looked at both issues, i.e. from an end user's and an administrator's point of view.

The work of the subjects comprised some initial product and market research in order to identify relevant test candidates and the setup of a testing environment. The test platforms mainly consisted of Windows 2000 or Windows XP PCs in small networks. The VPN team also used a Linux machine to run a FreeS/WAN server. The results can nevertheless be generalized to some extent since a significant part of the test candidates is also available on multiple platforms. For instance, Microsoft Internet Explorer is also available for Mac OS, the Apache web server or the Netscape/Mozilla clients run under all common Windows, Linux, or Unix versions etc. During the evaluation, the test subjects examined all the categories and central questions given therein. Results were summarized in written reports and finally presented in class.

3.4.3 Findings of the Study

In this section we summarize the main findings of the Microsoft usability study and point out conclusions that had found their way into subsequent research.

Benefits of PKI In its beginning the study clearly pointed out the need for PKI to face the risks of modern IT systems. The theoretical background of (public key) cryptography and public key infrastructures were then explained in detail (comparable to our treatment in Section 3.1). It was argued that the underlying concepts necessary for PKI are a mature technology as they had been known and researched by the scientific community for decades. A number of use cases had been identified where PKI provides a real added value as an "enabling service" allowing secure communication in situations where it would have not been possible otherwise. From an end

Use Case	**Candidates**
SSL-enabled HTTP Server & SSL-enabled Web Browser	Apache 2.0 (with integrated mod_ssl) Microsoft Internet Information Services (IIS) 5.1 Internet Explorer 6.0 Mozilla Navigator 1.3
Secure email (S/MIME-conformant)	Netscape Mail 7.0 Outlook Express 6.0 Outlook 2000 Pegasus Mail 4.11 (with pm-smime extension) The Bat! 1.62r
Groupware	Exchange Server 2000 (with Outlook client) Lotus Notes/Domino 6.01 (client+server) Novell GroupWise 6.5 (client+server)
File encryption and document signing	Abylon Protection Manager Pro 4.0 Microlog Cabinet Manager 2003 v4.1 Microsoft Office 2003 (signing only) RSA Security SureFile 6.0 SECUDE Office Security Suite 2.4 T/bone FileProtector 1.1 Vigilant 4.3.3 Windows XP Pro EFS (encryption only)
Token-based System Logon	KOBIL Smart Token (for Windows XP) T-TeleSec NetKey for Windows XP
VPN Server & VPN Client	FreeS/WAN Server 2.0rc1 Windows 2003 VPN Server SSH Communications Sentinel Client 1.4 Windows XP VPN Client
RADIUS Server & RADIUS Client	AEGIS Premium 1.1.3 Odyssey Server 1.1 Steel-Belted Radius 4.04 AEGIS Client 2.0.5 Odyssey Client 2.0 Windows XP Standard Client
Cryptographic APIs	Generic Services Security API (GSS-API) Java Cryptography Architecture (JCA) Microsoft CryptoAPI PKCS#11

Table 3.2: Applications/APIs tested in the Microsoft Usability Study.

user's viewpoint, the use cases significantly differ with respect to the level of transparency of the PKI features and the level of direct involvement into security tasks. While for instance secure Web browsing can happen in a rather seamless way, secure email still requires a considerable cognitive and operational workload.

Market Situation A pleasant finding of the market analysis was the fact that for each use case a sufficient set of programmes existed. These programmes provide built-in PKI base functionality in a suitable way and relieve users of installing it separately. Users that are anxious about security thus have enough alternatives to select their PKI-enabled application of choice. A lot of the applications are available free of additional costs as they are part of the operating system or open source software/freeware. The current version of the Windows operating system, for instance, provides basic client-side support for all of the use cases.

Product ratings Altogether, the product tests yielded a satisfactory result. For the most common use cases (e.g. secure Web browsing), our test subjects assigned an overall score of "good" or "very good" to the majority of products. They estimated the necessity of training as low for applications that target end users, but non-negligible for administrators that manage server applications. With respect to the design of security-relevant menus and dialogues, the test persons raised a number of issues. They criticized that configuration settings are not organized logically and properly and complained about the sloppiness of warning messages and error handling. In their opinion most of the applications lacked the possibility to adjust the transparency level. Among the open issues of most applications were the functionality for certificate lookup and validity checking. Standard protocols like LDAP or OCSP and standard mechanisms like CRLs and delta CRLs had often been found to be unsupported.

3.4.4 Lessons Learnt

In this section we summarize the experiences we got from applying our evaluation framework in practice. From the feedback we received from the test subjects during the evaluation and the quality of their final test reports, it can be said that our approach is indeed an effective and practical evaluation

tool. The organisation and categorization given therein had been considered a natural and logical choice, which covers all relevant aspects and provides clear guidance for evaluators.

The focus of the study (and especially our client) was to examine the potential of PKI-enabled applications that is why we developed a methodology that clearly falls into the category of inspection methods. Conducting laboratory tests with end users alone would probably not have been sufficient to satisfy this requirement and to provide an in-depth analysis of technical features as experiences from the field indicate [176]. Within the bounds of this study, laboratory tests would have also been by far too time-consuming and thus infeasible due to the numerous use cases and test candidates. In our opinion, the expert inspection method we chose allows a reasonably deep assessment of security features and usability pitfalls at an excellent cost-benefit ratio. In all, the study had been a valuable and successful testing ground for our evaluation framework.

3.5 Trustcenter Software Study

Apart from the Microsoft Usability Study, our framework has also been applied to carry out a thorough analysis of trustcenter software. Microsoft offers a built-in trustcenter component in its server system since Windows 2000. For the following section, the most recent and most powerful version, namely Windows 2003 Server Enterprise Edition, was evaluated according to the framework. We also had the rare opportunity to evaluate a full-fledged, high-end professional trustcenter software in the form of Entrust PKI.

The complete results of these evaluations can be found in the publication [151] and the extensive (about 100 pages) evaluation report [150]. Due to space restrictions we can only summarize them here as we did it in the case of the Microsoft Usability Study.

3.5.1 Test Candidates

The following two sections contain a brief technical description of the test candidates to give the reader an impression of the feature richness. Product descriptions can be found on the vendors' web sites[12].

3.5.1.1 Windows CA

The Windows 2003 Server CA or Windows CA for short is an X.509-based PKI solution that implements a bunch of PKI services. It supports a lot of use cases by means of pre-configured and extendible certificate templates. Templates do not only govern the X.509v3 certificate profile (for instance validity period, certificate contents, and extensions), but also specify policy issues with respect to the enrolment process, cryptographic parameters like algorithms, key length, and CSP (cryptographic service provider). The template mechanism is a powerful tool since its introduction in Windows 2000. It facilitates the management of the PKI user base. User enrolment automation is possible as Windows CA strongly integrates with Microsoft's Active Directory (AD) server and Windows domain controllers. This statement, however, holds only in a homogeneous Windows operating system environment. CAs can be operated online or offline and can be organized in multi-stage hierarchies and connected to other PKIs via cross-certification.

Windows CA originally supports the common cryptographic algorithms (RSA or DSA as signature schemes in combination with SHA-1 or MD5 and key lengths up to 16384 bit) and allows to load other algorithms via additional "providers" – a library function mechanism similar to those of the Java Cryptography Architecture/Extensions (JCA/JCE). The CA's as well as users' private keys may be stored either on PKCS#12 softtokens or hardtokens like SmartCards or USB devices. Windows CA supports X.509v2 CRLs as well as delta CRLs, which can be distributed by all common means (LDAP, HTTP, or via the file system), but does not include its own OCSP responder.

[12] `http://www.microsoft.com/technet/prodtechnol/winxppro/plan/pkienhancements.htm`, `http://www.windowsecurity.com/articles/Windows/Server/2003/Certificate/Services.html`, `http://www.entrust.com`

3.5.1.2 Entrust PKI

Entrust Limited, a Canada-based company, is one of the market leaders in certification services and trustcenter software. Entrust has offered PKI software since 1994 claiming to having sold the "world's first commercially available PKI".[13] In contrast to the Windows CA, this solution targets really large CA installations serving millions of users. Obviously, the Entrust PKI would typically be an oversized and overpriced product for a small company. The two products can nevertheless be compared as our requirements towards key and certificate lifecycle management are similar – Entrust focussing more on automation in large user populations and the Windows CA on the ease of deployment in a homogeneous environment.

The Entrust Authority Security Manager (version 7.0) as the core certification authority is the basis component needed to build up a PKI. It has quite specific operating system (several Windows server or Unix platforms) and database (Informix or Oracle) requirements. The Informix database is bundled to the Security Manager such that the PKI can be setup without additional third-party software. The minimal installation is completed by the "Entrust Authority Security Manager Administration" module, a graphical user interface for the Security Manager.

Entrust Authority implements a similar certificate template mechanism allowing flexible enrolment including automated enrolment and user self-services. All major PKI standards (X.509 certificates and CRLs), PKIX-CMP, PKCS#7, PKCS#10 and SCEP are supported. The software interoperates with LDAP directories, OCSP responders as well as SmartCards and HSMs (hardware security modules). Entrust Authority is EAL 4+ certified according to Common Criteria and successfully passed an ISIS-MTT conformity assessment.

3.5.2 Methodology

Before we explain our methodology for the evaluation of trustcenter software in detail let us note a few particularities of the test. Obviously, trustcenter applications are neither general user nor COTS software. As such they target a small user circle, say network and system administrators, corporate security coordinators or security consultants. However, end users are typi-

[13] `http://www.entrust.com/pki/index.htm`

cally also involved as they get into contact with the PKI when for instance requesting or retrieving certificates. Both perspectives have to be taken care of.

In addition to the heuristic approach, we broadened our methodology and added laboratory user tests. Those tests were based on two usage scenarios with an increasing number of constraints. These scenarios describe the situation in small and medium enterprises (SMEs); a description is given below. The test subjects acted in the role of system administrators that had been given the task of setting up an internal PKI for the company. We believed that proceeding along the lines of a scenario is an appropriate way to focus on a realistic subset of a trustcenter application's functionality. Besides, the complexity of the subject usually requires to pre-configure an appropriate environment in order to be able to access certain dialogues of the application (e.g. attach the application to a domain controller or directory service in order to assess certificate publishing facilities). It thus seemed more natural to restrict oneself as in practice one would typically also use only a fraction of the whole features. Anyway, we had to cope with limited resources for the evaluation, so the two SME scenarios were what we could cover realistically.

3.5.2.1 Test Scenarios

We formulated two scenarios, a relatively small one and a more complex one. The latter has a superset of requirements compared to the former one. This choice is not only advisable due to the complex nature of the subject, but also reflects the way one would work in practice. Typically, a pilot phase of manageable size, where tests can be made in a controlled environment, would precede a large, company-wide system installation.

Scenario I reflects the requirements of a small company, having 20–100 employees, where all of them work at the same location. We assume the protection of email with customers as being the dominant use case in this setting. Companies of this size are likely to rely on standard software, so we assume a homogeneous environment where all machines have the same Windows operating system and use Outlook Express and Internet Explorer[14] as clients. We deliberately state only very few constraints. The motivation

[14]The choice of the web browser in fact matters as some PKI tasks are browser-based, e.g. client-side key pair generation.

is to model an administrator whose idea of the final shape of the PKI is a priori not too clear and to allow some degree of freedom.

For the second scenario we assume a larger user base and take into account that the employees are spread over different locations. As a consequence, requirements towards key and certificate management, auxiliary services (e.g. with respect to the need for some form of automation), and the security level in general grow. It should be possible to use different key pairs for signing and decryption. A backup of decryption keys should be made by default, but copies of signature keys must not exist. We also demand that there are certain users in the system for which key backup is deactivated. Key backup in general is a desirable feature as to prevent data loss, but there are sensible environments, for instance think of the R&D department, were key backup violates the observance of secrecy. The heterogeneity of the application side is reflected by the choice of MUAs and web browsers.

We chose beforehand not to consider hardware PSEs as interoperability and driver issues have often shown to be a significant technical obstacle with PKI in the field [126]. Details of the two scenarios are listed in Table 3.3 and 3.4.

3.5.2.2 Adaptations and Extensions

As already mentioned, we combined an inspection and an empirical method for the evaluation of the two trustcenter applications. Consequently, we also slightly adapted the evaluation categories to the particular situation. First of all, the category "reaction in security-critical situations" was dropped in favour of the scenario analysis as we could treat that issue in much more detail there. We outline the particularities of each evaluation category in the following.

Gathering Information and Obtaining the Software Trustcenter software is not very widely deployed or used, so it is often hard to get appropriate information especially with regard to standards compliance and the support of a particular intended usage scenario.

Technical Requirements With respect to hardware: mostly a question of performance as we did neither use HSMs nor SmartCards nor USB tokens. With respect to software: restrictions towards the operating

- Users: 10 employees at a common location.
- Use Case: secure email with customers (encryption and signatures).
- Keys: only one key pair per user, stored in a softtoken.
- Client OS: Windows 2000.
- Applications: Outlook Express, Internet Explorer.

Table 3.3: Scenario I for trustcenter evaluation.

- Users: 100 employees at three different locations.
- Use Cases: secure email (internally and externally), encrypting and signing files.
- Keys: two distinct key pairs, one for encryption and one for signatures, both stored in softtokens.
- Client OS: Windows 2000
- Applications: Outlook Express, Thunderbird, Internet Explorer, Firefox, Opera.
- Additional requirements: key backup/key recovery should be made possible for certain user groups and certain keys respectively. Separation of duties for administrative tasks according to a reasonable role model. The CA component should reside on an offline machine.

Table 3.4: Scenario II for trustcenter evaluation.

system (version, build number, language[15]) and components like the directory.

Installation and Configuration During the setup of a trustcenter, installation options and design choices at an early stage may affect the system's functionality and options with a certain delay. The definition of certificate profiles or the decision whether to operate the CA online or offline are two examples. In order to avoid dead ends the software should either accompany the installation by a wizard or avoid such determining.

[15]For instance it turned out that Entrust Authority does not work with a German version of the OS although stated otherwise.

Technical Support We actually could not assess the quality of the vendor's individual hotline or email support. Instead we resorted to product-related man pages and the online knowledge base.

Training Similar to the previous category, product-specific trainings could not be assessed. We estimated the amount and type of training which seemed necessary against the background of the dialogue design and the available documentation.

Menus and Dialogues Trustcenter software is clearly a niche application. As a consequence there are no established best practices for interaction design in this field (as opposed to for instance secure email or secure web browsing). Using a precise and understandable language is crucial.

Transparency of the Security Features The products under test allow a myriad of settings. In order to not let users get lost, the application should hide complexity in an appropriate way and provide reasonable default settings that can be refined later.

Warning and Error Messages There is a typical phenomenon with security software in general and trustcenter applications in particular: Warning and error messages do not provide clear instructions of how to react or do not point out alternative ways when the user is stuck.

Help System Especially in the course of the installation, a user may easily get lost in the details and lose track of things. Therefore, the help system should contain sample scenarios or step-by-step guides (e.g. in the form of a check list) for complex tasks.

Reaction in Potentially Threatening Situations Category is treated separately (see above).

Algorithms and Parameters The application should provide some clues in judging the appropriateness and strength of algorithms. This is important with respect to long-term security and directly influences the choice of certificate validity periods.

Secret Key Handling On the one hand, this issue is relevant for the CA keys (usage of hardware PSEs). On the other hand, it also applies to end entities keys and the possible transport of software PSEs to end users or networked machines during the enrolment phase.

Certificate Management A trustcenter potentially has to cope with large numbers of users/certificates over a long time. Therefore, it should be feasible to ease or automate the process of certificate lifecycle management as much as possible. For instance this includes certificate renewal notifications or batch processing.

Status Information For the special case of trustcenter software we widened the meaning of status information to also capture the application's log files in order to allow an independent auditor to assess the security of the PKI. This includes for instance key history and archival.

Advanced Functionality There are a lot of potential specialisations here. We only name two of them which were important for our evaluation scenarios. One of them is the enforcement of security policies, e.g. with respect to role-based access control. Implementing a proper password management and password "hygiene" is also crucial as administrators often do not authenticate via public key methods, but use passwords.

3.5.3 Evaluation Reports

In the following we summarize the course of the evaluation of both products. For the Windows CA, we restrict ourselves to the first scenario as this overlaps to a considerable extent with the second one. Entrust Authority was evaluated by the three test subjects working in group. Since Entrust has high and very specific hardware demands for Scenario II (two additional machines for the installation of a second CA and a domain controller respectively as well as one machine for Entrust's Enrollment Server), we restricted ourselves beforehand to the evaluation of Scenario I. We refer to [150] for a complete and much more detailed description.

3.5.3.1 Windows CA

The installation process of the software can be subdivided in the steps shown in Figure 3.5. At the very beginning the test subjects had to chose between one of four possible CA types: "enterprise root CA", "enterprise subordinate CA", "stand-alone root CA", and "stand-alone subordinate CA". The choice determines to a large extent the shape of the coming PKI hierarchy.

The test subjects were somehow helpless and indecisive which variant to select. They all rated the explanation as incomplete and misleading as, for

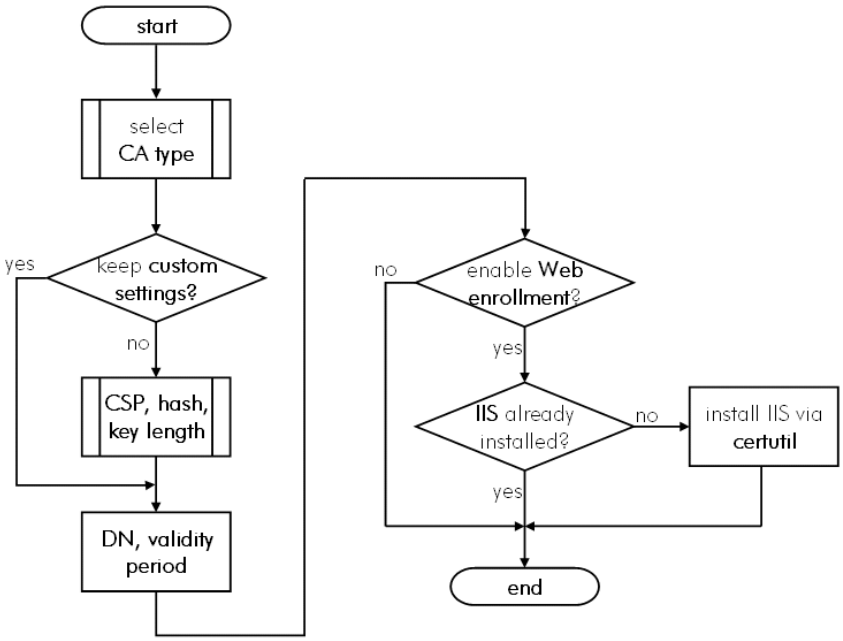

Figure 3.5: Installation process of the Windows CA.

instance, the "enterprise root CA" is characterized as the "most trusted CA in an enterprise" while the "stand-alone root CA" is described as the "most trusted CA in a CA hierarchy". The installation wizard did not provide further information nor a context-sensitive link to the documentation.

Two of the testers finally settled on the enterprise and one for the stand-alone root CA. In their first run of the installation, they all decided to accept the default setting for the CA's key including signing algorithm and hash function, key length and Cryptographic Service Provider (CSP). Nevertheless, they complained about the lack of feedback as the system defaults were not visible. When playing around with the custom settings in a later run, they missed a wizard that recommends certain combinations of algorithms, key lengths, and CSPs or at least checks them for consistency. The remark "individual CSPs may have different cryptographic algorithms that provide various levels of security" completely confused them. A checkbox labelled "allow this CSP to interact with the desktop" and the insufficient information in the documentation ("Without this option, system services cannot interact with the desktop of the user who is currently logged on.") made one of the test subjects intensively search the Web. He found some clues that

this option could have something to do with PIN entering for SmartCards, but still wondered whether it was about a security-critical setting.

The testers chose the default settings concerning the CA's distinguished name (DN) and certificate validity in the next dialogue. Later we recognized that the application neither checks the DN for consistency – e.g. it was possible to include two country attributes ("CN=TestCA, O=TU Darmstadt, OU=CDC, C=DE, C=FR") , which could confuse PKI-enabled clients – nor aligns the certificate lifetime with the key length. Even a lifetime of 1000 (!) years could be selected without warning. The initial key length also affects the renewal of CA certificates as the length can only be increased for keys with less than 1024 bits, which the testers found incomprehensible. Last but not least, none of the dialogues informs the user which signature and hash algorithm is actually used.

Up to now, none of the test subjects had installed Microsoft's IIS web server as they had been given no appropriate hint. Consequently, they were surprised to encounter a warning that web-based certificate enrolment will not work. One of them thought that it would be necessary to install the CA anew preceded by the IIS installation. The second one tried to configure the web interface via the command line tool certutil while the third did not care about web enrolment at all. The three testers praised the CA's configuration menu (see Figure 3.6) as well-understandable and comfortable to use. The menu offers to set for instance CRL distribution points (i.e. publication locations for CRLs), the internal roles of the CA, and the policy module. After modifying the policy in a way that certificate requests have to be confirmed manually, all three were ready to request sample certificates on behalf of an imaginary user.

Basically, there are two ways to request a certificate, namely via a so-called snap-in of the Microsoft Management Console (MMC) or via the web interface (if installed). Users are authenticated using their Windows login credentials. In both cases, the user may post new requests, view the status of pending requests, install personal certificates, and download CA certificates and CRLs. The test subject who had chosen a stand-alone CA had to enter the subject name and email address in the web interface as the CA was not connected to an Active Directory. Prior to that, he had to activate ActiveX controls for Internet Explorer, which he found cumbersome as this step requires individual configuration of each client with administrator privileges.

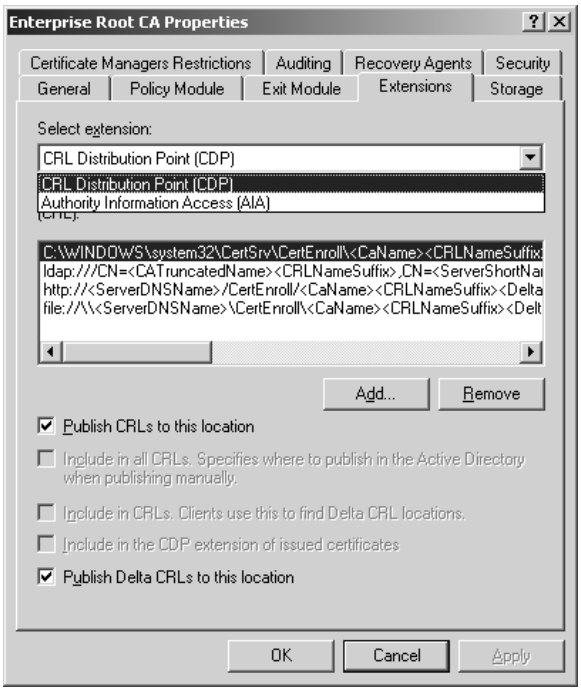

Figure 3.6: Main Windows CA configuration menu.

The other two subjects created their key pair and certificate signing request via the MMC, which they considered a tool unknown and unfamiliar to most of the average Windows users. In the course of the evaluation of Scenario II they learned how to restrict the requesting person's alternatives concerning CSPs via certificate templates. They were surprised to see that this presetting did not influence the MMC dialogues (however, it works properly for web-based certificate requests). This results in unintelligible error messages if the end user selects an unsupported CSP.

Scenario II required an offline CA, i.e. a system that is not connected to the company's network let alone the Internet. The test subjects had to look for workarounds as they could not figure out a proper way to solve the problem. Microsoft proposes to use a hierarchy composed of an offline "stand-alone root CA" plus a subordinate CA that manages all requests, but is online again which the testers regarded as counter-intuitive. Furthermore, they considered this a dead end because it does not match the scenario's requirement. None of them came up with the idea to try issuing end entity certificates with the root CA.

One workaround consisted in setting up the CA including web front end (as the web pages could not be installed separately) and then deactivating the CA service, but keep the IIS running. The rational behind this try was that the key pairs are generated locally in the user's web browser. However, this approach did not work. The second workaround was to use a dummy CA that accepts incoming requests, but does not certify them. Instead, the request should be transferred manually to the actual CA by the CA administrator. Two alternatives were explored, namely saving the request via the web interface or retrieving it from the "pending requests" folder. Certification indeed worked with the first variant, however, the certificate subject got changed to the identity of the administrator. A similar phenomenon was noticed in the second variant although one of the testers reassured himself that the intermediate request file still contained the correct value. To cut a long story short, the test subjects did not succeed in setting up an offline CA in the sense of the literature.

For Scenario II the testers experimented with certificate templates. It took them some time to figure out that the unleashed power of the template mechanism could only be used in the *enterprise edition* of Windows 2003 Server, not the *standard edition* they had on their machines. Only one person found some hints in the documentation regarding the differences between the two versions. For instance, new templates could be defined, but not applied in the standard edition – a restriction that yielded much confusion and frustration as the whole operating system had to be installed anew.

3.5.3.2 Entrust PKI

The first impression the test subjects had of the Entrust PKI was the loads of documentation, which was shipped with the software (more than 2000 pages in total). These documents treated particular aspects of the CA installation, however, the testers were unable to structure the material and especially find out the order in which the documents are needed for the installation. They commented that – because the module names sound similar and not very descriptive – it would have been helpful to have a central place where the relevance of the separate modules, their functionality, and links to the corresponding documents is listed. As they did not find clear guidance concerning the order of the installation, they determined it manually on

the basis of the installation prerequisites listed in the various documents. To sum it up, it can be said that an intuitive approach (without extensive reference to the documentation) failed, because the wizard does not provide sufficient guidance. Among other fatal errors that occurred, the installation routine did not point out that the CA cannot interoperate with a German-language version of Active Directory (see below).

Disk space required for...	Default location	Formula
audits	/authdata	A = U*P, expressed in Gbytes
database	/Ifmxdata	D = U*(2+2*I+2*Y*R), expressed in Gbytes[1]
transaction log volume between backups	transaction logspace	T = 0.1*(L)/10,000, expressed in Gbytes (0.1 Gbytes per 10,000 update operations)
online backups	/entbackup	A+D+T, expressed in Gbytes

Figure 3.7: Entrust's formula to calculate database size (taken from the manual).

Entrust uses an Informix database for internal purposes. Its installation did not pose major problems given the small number of users in the test scenario. Nevertheless, the dialogue window responsible for the database configuration was criticized because of the cumbersome method the database space has to be calculated by hand (Figure 3.7 shows an extract from the manual). It remained unclear whether the database size can be increased later on.

The testers praised that Entrust Security Manager allows to control hardware requirements and, in principle, allows to start by choosing one of several installation profiles. Unfortunately, they felt that the descriptions were vague and complained about the lack of a context-sensitive help. Having chosen the "simple configuration using Microsoft Active Directory" they were able to continue, but were not informed about the signature algorithm or the key length until the CA certificate was produced. The default selection of an RSA key with 1024 bit is at the lower end of what should be used today.

In the following major usability flaws and severe UI bugs were detected by the test subjects. When they used the "next" and "back" buttons to navigate through the Security Manager's configuration dialogue, previously entered data got lost or could not be modified again. In one case a complete

new installation was necessary. The wizard did not inform properly about complexity rules for the various operators' passwords. The testers found it astonishing that passwords had to be set via a command shell which – to top it all – crashed after three incorrect password entries (see Figure 3.8 for a screenshot).

Figure 3.8: Entrust security manager password dialogue.

After they had successfully accomplished this task, the software reported an error telling that the Security Manager could not be initialised and referred to the log files. Assuming that they had mis-configured something, the testers restarted the configuration steps with varying settings, but without success. An analysis of the installation log made them intensively search the web, Entrust's support web pages, the product documentation, and the CDs shipped with the software. Finally, they found a tool on a CD to administer the Active Directory in a way that it could interoperate with the Entrust CA. Making the AD fully compatible with Entrust turned out to be a complicated process as directory schemas and attributes had to be modified. As the testers were inexperienced with LDAP, they demanded for more detailed explanations and guidance. This is especially important against the background that a directory mis-configuration may bring other networked applications relying on the AD to a standstill. To make things worse, some configuration changes are irrevocable. As already mentioned above Entrust does not support non-English versions of Active Directory. It took the testers hours to track obscure error messages back to this constraint. The problem is that obviously some DN components (like "cert publishers" which is "Zertifikatsherausgeber" in the German AD version)

were hard-coded in the software. At this point the test had to be interrupted and restarted with an appropriate AD.

The Security Manager administration tool allows to manage certificate types (i.e. certificate profiles), user groups, policies, and audit logs from within a single interface. The UI and its feature richness were clearly oversized for the given scenario, a context-sensitive help or wizard support is missing. Certificate types can be assigned to persons either on the basis of group membership or individually. The testers wondered which of this alternatives would have the higher priority if both were conflicting.

Having set up the CA components, the test subjects set about testing the trustcenter software from an end user's point of view. Entrust Authority scans the Active Directory for new users. An administrator may manually add persons to certify. For each of them, a random 12-character activation code is generated, which the future certificate holder has to provide as authenticator during enrolment. This activation code is accompanied by a 8-digit reference number. Both pieces of information had to be transferred to the user in a secure way. According to the testers' opinion, Entrust does a fairly good job in pointing out the security level of the different transport alternatives (personal contact by phone or during a conversation, paper mail, electronic mail, or by a setup file, which can optionally be password-protected).

The web site which users have to access in order to enrol is shown in Figure 3.9 (when using Internet Explorer). It seems a common practice to make available the CSP detail settings to the end user (which the testers already criticized as both superfluous and error-prone in the Windows CA evaluation). A few issues were carped about. Firstly, some of the CSPs listed were unavailable. Secondly, weak providers that only supported RSA keys up to 512 bit were included. Such CSPs could in fact be selected by the end user clearly violating the policy previously set by the administrator that dictates a minimum key length of 1024 bit. Neither the user nor the administrator at least got notified of this problem.

3.5.3.3 Results

In this section the results of the evaluation according to the framework are summarized. In Table 3.5 the points for each category are indicated. Observe that we use the symbols $--, -, \circ, +, ++$ representing the integers from

Figure 3.9: Entrust Authority web frontend.

−2 to 2. The rating scheme keeps to the one described in Section 3.3.1.2. The categories were assigned a weight of 5% or 10%, the respective value is listed in the second column. Detailed justifications for the individual scores can be found in [150, 151].

To sum it up, the overall scores of both products are pretty close to one another (Windows 2003 Server CA: +0.3; Entrust Authority: +0.5). This may look surprising at first sight. One reason for this is that the weighting for the small and medium enterprise scenarios was done according to what we called the "enterprise" setting in Section 3.3.1.2, where security features do not play an outstanding role. If more importance is attached to this category group, the overall picture will change as Entrust Authority is clearly leading here. In order to recalculate the score in the "high security" setting, it would be necessary to fix new weightings for the individual categories, too.

The results have to be taken with a grain of salt since the focus of the two test candidates is different. While the Windows CA in fact targets SMEs, the solution of Entrust would be definitively oversized for this use case. The comparability is somehow limited by the variation in the test methodology (cf. Section 3.3.1.3). However, some interesting tendencies were found during the evaluation.

The Windows CA distinguishes itself by the only slightly below-average usability and the GUI design and mechanisms that resemble the familiar

Deployment		**Win CA**	**Entrust**
Information	5%	∘	∘
Technical Requirements	5%	−−	−−
Installation & Configuration	10%	∘	−
Technical Support	5%	∘	−
Training	5%	+	+
Ergonomics			
Menus & Dialogues	5%	+	−−
Transparency	10%	−	−
Warning & Error Messages	5%	−	−
Help System	10%	∘	−
Security Features			
Algorithms & Parameters	5%	+	++
Secret Key Handling	10%	+	+
Certificate Management	5%	++	++
Status Information	5%	−	++
Advanced Functionality	10%	+	++

Table 3.5: Results of the framework evaluation.

ones of the server operating system as a whole. As in this respect the CA is an additional feature only, we expected a limited functionality, but the evaluation set us right. The time needed to install and put the CA into operation, however, was quite high. The test subjects needed approximately 5 workdays for Scenario I and 14 workdays for Scenario II. This sounds a lot, but one has to take into consideration that the test participants were no professional and experienced Windows system or network administrators. Nevertheless, this number is not too far away from Microsoft's own estimation of 100 hours (sic!).

The Entrust CA made a bad impression as it over-emphasizes security features to the disadvantage of ergonomics. The significant usability deficits may quickly have a negative impact on the set-up costs of such a CA installation. In our opinion the investments in security do not justify to neglect usability because the user errors we encountered during the evaluation may lead to security problems again. The time frame which was originally planned for the evaluation of Scenario I was exceeded by far with no less than 24 workdays. Therefore, we refrained from extending the requirements according to Scenario II. A resumption of the evaluation on the basis of the present findings would nevertheless be possible. As a next step, a broader

test of Entrust's rich features, that go beyond Scenario II would be desirable and should be feasible with our framework.

A number of weaknesses was found during the evaluation (see [150] for details). Although the systematic search for usability deficits or security flaws was not in the main focus of the evaluation, this is more than only a by-product. Weaknesses were classified according to their severity and grouped logically according to the category groups of our framework.

3.6 Conclusions

After giving a brief introduction into the main concepts of public key cryptography and PKI, we identified a number of challenges on the technological layer, which are a major source for usability problems with PKI-enabled applications. Among those challenges are the management of keys and trust anchors as well as the delegation of private keys, which one could consider open problems. We present solutions to two instances of these problems, namely opportunistic security for email and delegation of WWW credentials in Chapter 6 and 7 respectively.

The analysis of technological challenges also is the foundation for our new and generic framework to evaluate the usefulness of PKI-enabled applications. It is not restricted solely to usability, but also treats deployment issues and security features to get utility ratings, too. Our PKI evaluation tool can be adapted to a multitude of PKI application families. In the context of the Microsoft Usability Study the framework has proven to be flexible enough to fit to a wide range of use cases – ranging from secure web browsing to the assessment of cryptographic APIs. The design of the framework is technology-agnostic, use case-neutral, and evaluation methodology-independent. As we show in Chapter 7 it can also be used as a tool for application development.

We showed how to use the framework for the evaluation of trustcenter software, which is possibly the most complex example of a PKI-enabled application. We combined a heuristic evaluation with an extensive laboratory user test. Surprisingly, the test candidates suffer from elementary usability flaws, which decreases the confidence in their functionality, e.g. when crashing due to false inputs. The products also did not check security-relevant settings for consistency and plausibility (key length, certificate validity).

Although the test participants were students with some basic experience in the administration of Windows server and with deeper security and PKI knowledge, it took them intolerably long to complete the laboratory tasks (approx. one and two weeks for the Windows CA and more than three weeks for the Entrust PKI only with the simpler scenario). It would be desirable to cross-check these findings with actual system and network administrators. Microsoft estimates at least 100 hours to set up its trustcenter product, which in the light of our evaluation, does not seem too pessimistic. Another idea is to extend the framework to some kind of "usability test bed" by formulating precise scenarios for each application family or use case. This test bed could serve as the basis for further analyses.

Part II

Solutions

Chapter 4

Awareness by Doing

> *O negligence! ... what cross devil made me put this main secret in the packet I sent the king? Is there not way to cure this? No new device to beat this from his braints?*
>
> – William Shakespeare

> *The human side of computer security is easily exploited and constantly overlooked. Companies spend millions of dollars on firewalls, encryption and secure access devices, and it's money wasted, because none of these measures address the weakest link in the security chain.*
>
> – Kevin Mitnick

Experience has shown that information security incidents can often be traced back to human misconduct or misunderstanding. Technical measures seldom can solve the security problems alone. The goals of so-called *information security awareness* programmes or "campaigns" are to attract attention, to gain and maintain users' interest, and to impart basic and practical knowledge. In the following, we will often write security awareness or just awareness for short. The importance of security awareness is accepted among IT managers and backed by relevant surveys (see below). It has also become an important part of the curriculum for IT security consultants, e.g. in the CISSP (Certified Information Systems Security Professional) or T.I.S.P. (TeleTrusT Information Security Professional) education [232].

In this chapter, which is based on our paper [16], we propose our new approach of Awareness by Doing, which involves users more directly and in-

tensively into the awareness programme in order to achieve a substantial and lasting effect. In Chapter 2 and 3 we saw that PKI is a complex matter, not only from a technological, but also from a cognitive point of view. It is crucial that users are aware of threats and know when to be suspicious and how to act properly. This holds for IT security in general and for cryptography- and PKI-based mechanisms in particular. We already mentioned the challenges of validating certificates and fingerprints or human factor issues with SSL (cf. Section 3.2 and 2.2.2.3). On the one hand, the approach described in this chapter is the continuation of the previous analysis and experiences gathered from user studies as it shows how to translate the ideas into concrete action. On the other hand, our approach also incorporates elements of laboratory user testing and may therefore give a fresh impetus to the study of security mechanisms, e.g. in the form of a field study.

We begin with a description of the status quo providing some recent figures and continue with a review of related work and activities in this area (Section 4.1.2). As our approach falls back on findings from the domain of experiential learning, we summarize these prior to the presentation of our methodology. We also provide guidelines how to organize an Awareness by Doing campaign and how to measure its success. At the end of this chapter, some sample learning modules are described to illustrate our approach.

4.1 Status Quo

4.1.1 Some Figures and Insights

An Ernst&Young survey [75] asked companies which threats to IT security they perceive as being the most important ones. Unsurprisingly, viruses and worms are at the top of the list (77%), but employee misconduct was already second, being relevant for more than half (57%) of the respondents. Despite these figures companies still have a very strong focus on technology when planing security measures and pay too little attention to employee awareness or education programmes. Only 29% of organizations list employee awareness and training as a top area of information security spending, compared with 83% of organizations that list technology as their top information security spending area – a finding that is backed by the analysis of [254]. This study concludes with the observation that “relegating human resource concerns to such a low priority may be seriously shortsighted, in view of the

potential threat and costs" as a greater security benefit could be achieved by intensifying efforts on human capital issues.

The recent kes/Microsoft "Lagebericht zur Informations-Sicherheit" (status report on information security, [142]), an in-depth review of the status quo of information security in Germany, came to similar conclusions. The companies taking part in the survey identified their own workforce as the #1 source of threats to the company's security. Human error and carelessness are by far the most important causes for endangering potential. More than half of the companies traced concrete incidents and damages back to these phenomena. The findings are very similar to the preceding study [245] two years before. The situation is not likely to improve, staff error is still leading among the danger areas (cf. Figure 4.1).

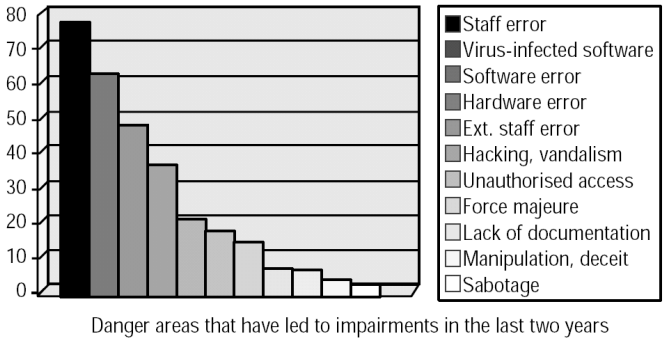

Figure 4.1: Finding of the kes security research study (taken from [254]).

The idea of awareness however is not a restricted to staff members, but also has to find its way to the minds of senior executives [240]. The biggest hurdles to an effective improvement of IT security (apart from budgets, which are always too small) is in fact the lack of awareness on every level in the enterprise hierarchy [142, 113]. Of course, IT security awareness is neither restricted to enterprise users. New research from the UK has found out that 83% of the population do not know enough about protecting themselves online [240]. Interestingly, to 17% of the people Internet crime is of greater concern than physical crimes like car theft and mugging.

4.1.2 Related Work

Requirements and components of a successful security awareness training are worked out by the American NIST (National Institute of Standards

and Technology) in [177]. There a three-step approach is proposed: As an introduction, users' attitude should be changed such that they recognize the importance of security measures and possible consequences of security breaches. The actual knowledge transfer is the second and the most extensive step. The purpose of training courses is to enable users to complete their working tasks more efficiently and securely. Finally, the education of IT security professionals ("train the trainer") is outlined.

Based on his rich practical experience, Fox [84] describes a phase model for the organization of a security awareness campaign in an enterprise: In the first stage, "marketing" techniques like circular letters or flyers should be used by the management to gain attention for the campaign and underline its importance. The actual knowledge transfer which aims to influence users' attitude is the subject of the second stage. Examples include informational meetings, automated web-based trainings, or the provision of documentation and reading materials. As a further fortification, Fox mentions media coverage by means of internal news letters or publicity gathering actions like a lottery (stage 3). Depending on their PR strategy, companies may chose to communicate their awareness activities and the goals achieved towards partners, customers, or even the general public in the final stage of the programme. Examples of successful awareness campaigns can be found in [85].

There are various products and events available on the German market today which aim at improving awareness. The CD-ROM "In's Internet – mit Sicherheit" ("Access the Internet – securely") published by the German Federal Office for Information Security (BSI) or the BSI citizen portal[1] are good starting points. BSI's Internet course "Baseline IT Security" covers a wide spectrum of topics – including awareness and information security education and training. It ranges from risk management and asset classification to physical and environmental security. This course lets people proceed step-by-step or delve directly into single modules. *CrypTool*[2] is – according to its self-description – a "demonstration application for cryptography and employee awareness". It is more targeted towards advanced users like students or application developers than towards the main body of computer users without prior knowledge in the area of cryptography.

[1] `http://www.bsi-fuer-buerger.de`

[2] `http://www.cryptool.de`

IT security awareness is paid special attention, for instance by the University of Tennessee, which has its own Information Security Office[3]. It provides a couple of short awareness videos, which users can view in their web browser, covering topics from virus protection to firewalls. The UK government in cooperation with major national banks, IT and telecommunication companies, has recently launched a large campaign to improve UK citizens' knowledge about online security. The *Get Safe Online Campaign*[4] targets both end users and small businesses. It provides in-depth information at an online portal, risk assessment questionnaires, and web-based training guides. What is interesting about this campaign is the fact that it is not limited to online media, but also includes large-scale roadshows and further events.

Lecture series and symposia are useful add-ons to an awareness campaign. There are annual lectures series of the Darmstadt Centre for IT Security (for instance "Secure computer usage in the era of viruses, worms, and misuse of electronic debit cards" in the summer semester 2004 in cooperation with the BSI), or the monthly workshops of the Competence Center for Applied Security Technology (CAST) and, last but not least, the annual awareness symposium by Secorvo Security Consulting GmbH.[5] User awareness can only be achieved if the operators of IT systems are familiar with the subject of IT security. A lot of universities offer lectures, seminars, and internships as their standard programme. IT security is also on the agenda of vocational education: The *Kompetenzzentrum IT-Bildungsnetzwerke*[6], a joint project of the IG Metall trade union and the association BITKOM (Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.), offers a course CD including examination questions for apprentices.

4.2 Our Methodology

In this section we motivate and explain our methodology. A short excursus will also lead us to the subject of learning methods – this analysis will prove

[3] `http://security.utk.edu`

[4] `http://www.getsafeonline.org`

[5] See `http://www.dzi.tu-darmstadt.de`, `http://www.cast-forum.de`, and `http://www.secorvo.de`.

[6] KIBNET, see `http://www.kibnet.org`

useful in estimating the effectiveness of our concept. In the main part of this section the organization of a campaign according to the Awareness by Doing methodology is outlined.

4.2.1 Rationale

Besides technical and organizational protection mechanisms, user awareness and user training are two important pillars of information security. As we have argued above there is a constantly growing pressure to act on this field, because today investments in IT security measures are still concentrated mostly on technological aspects and thus neglect user sensitization and training.

Too few companies comprehensively engage in security awareness today. If they do, they still rely on classical teaching instruments for knowledge transfer. Among the tools which are used, are written materials in the form of an IT security "vademecum" that describe data protection and data security guidelines, computer-based multimedia courses (possibly combined with online self-tests), face-to-face trainings, and seminars [84, 254, 198].

Seeing is Believing Our hypothesis is that these measures are not sufficient. Computer users in general are slow in changing their habits and they often do not achieve a lasting effect. This conjecture is backed by a CERT/CC estimation[7] that 80% of all network security problems are due to weak passwords – a thing which users totally have on toast. Although most users know the meaning of passwords in an *abstract* fashion pretty well, one can often find that they are rather careless about the way they use them in practice [202]. Users do not believe they are personally at risk – unless something happens to them [250]. A similar phenomenon can be found in the context of email: Although many people may be aware of *potential* eavesdropping on their email communication, they may dismiss that as improbable or they may be unbothered by it (see Section 2.3.6). Therefore key components of our approach are so-called penetration tests and a demonstration centre, i.e. a computer laboratory, where the working environment of users and possible attacks can be simulated in a realistic fashion, which comes close to their own experience.

[7] SecurityStats Password Security, see `www.securitystats.com/tools/password.asp`

Health – a Useful Analogy A telling analogy is due to Siponen [214] who draws the comparison between security awareness and the way humans deal with their own state of health: People do not worry about it very much as long as things go well. However, if something is going wrong, people suddenly have a strong interest in it. The trouble is that often, once you are sick, a lot of effort is necessary to recover completely from your illness – if this is possible at all. Let us stick to this point and the analogy for a moment. A good example for a successful awareness campaign are the efforts to fight AIDS/HIV, which started in Germany in the mid-80s and is still going on to date. The assessment of the effectiveness of these campaigns is the subject of annual polls which analyse people's knowledge, attitudes, and protection behaviour [45]. The education about AIDS is exemplary in many respects – take for instance the "branding" with its high visibility of the "mach's mit" ("use a condom") logo and motto on posters, in TV spots etc. But the campaign also shows that one must not slacken the efforts and eagerness: Infection rates are climbing again, indicators obtained from the polls become negative, for instance the perception of AIDS as a dangerous illness or the use of information offers. We believe that studying the results gained and the methods used in this area surely provide insights for the planning of information security awareness campaigns, too.

4.2.2 Goal

A security awareness campaign should result in an effective and lasting change of behaviour in order to minimize the exposure to security threats. Users should be taught to act in critical situations in such a way that minimizes the risk for themselves and/or their company.

Motivating Users The motivation users have at the beginning of the campaign has been identified as a crucial factor for the success of the programme [198]. In our opinion, it should be clearly pointed out already in the motivation phase that the user can also personally benefit from the security awareness campaign. Most users have IT systems in their private environment (e.g. home computers, mobile phones, PDAs etc.). An increase in users' security awareness and a basic understanding of what is the correct behaviour in security-critical situations thus also protects users' own IT systems. This reduces the risk for users of personally suffering from financial

loss, e.g. by becoming the victim of a phishing attack. The other way round enterprises also benefit if their employees are aware of security risks on their personal side as the attack vectors are often similar. For instance think of the protection against "shoulder surfers" that spy out PINs when users enter them at an ATM. This attack also applies to login passwords or the PIN of a SmartCard.

Direct User Involvement The key idea of our concept Awareness by Doing is the sensitization of users. Users should behave security-adequate and show a healthy measure of mistrust. They should be capable of distinguishing harmless situations from security-critical ones and know how to react appropriately in the latter case. Our concept aims at yielding more effective and more lasting results than previous awareness programs. In order to achieve this goal the user should play a role as active as possible and experience dangers first hand instead of only passively consuming information. We are not aware of similar approaches that emphasize direct user involvement as we do.

This approach is comparable to the way children acquire knowledge, which is an example of *learning by doing*. Typically children learn something and change their behaviour by exposing themselves to a certain situation and by experiencing the consequences of their actions immediately. A child learns that hotplates or ovens are hot and therefore dangerous by burning its hand. The feeling of pain illustrates the significance of the property "hot" much better than a thousand words of its parents.

4.2.3 Learning Techniques

In this section we make a short excursus into the subject of teaching and learning methods. There are two good reasons why one should apply different didactic and pedagogic methods in an awareness programme: On the one hand people have different "learning types" which – roughly spoken – means that the way they learn differs and that their method of choice is more effective than the others. On the other hand, the combination of different learning methods provides a higher and longer lasting result [147, 85]. The latter is a very important point since our concept emphasizes the lastingness of the contents in the awareness programme.

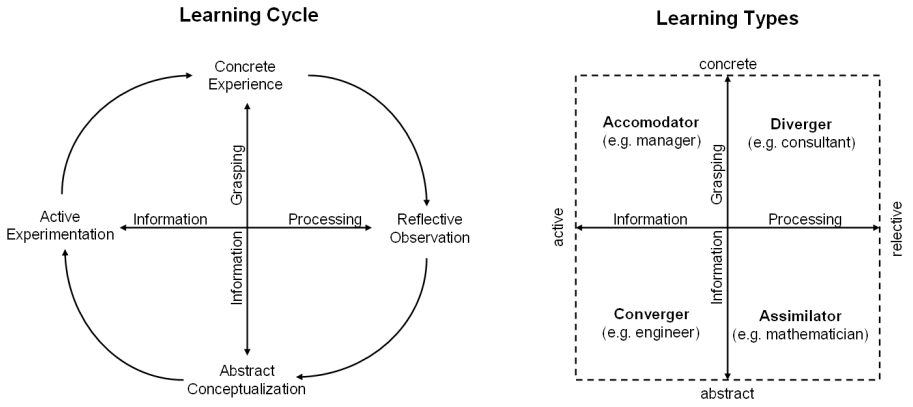

Figure 4.2: Kolb's learning cycle and learning types [147].

Kolb's Model In the following we give a brief introduction into the classification of learning types according to Kolb. We refer the reader to [147, 220] for an introduction. Kolb's model of *experiential learning* identifies four learning types on the basis of two criteria. These criteria refer to the way of how information is collected and on its subsequent processing respectively. Distinctions are made by using pairs of antonyms. In the first case, the pair *concrete experience* (CE) – *abstract conceptualization* (AC) is used to describe the extremes with respect to information grasping or perception. With respect to information processing, the pair consists of *active experimentation* (AE) of a concept in a new situation as opposed to *reflective observation* (RO). This distinction implies that, for instance, abstract or reflexive methods are not very well suited for people who prefer learning in a more concrete way and by experimenting. This statement especially holds for a subject that is as complex as IT security.

The four types, with a typical profession for each type, are depicted on the right side of Figure 4.2. Research shows that the population is more or less equally distributed among the four learning types [220]. It is reasonable to assume that this is also true for the participants of an enterprise-wide awareness programme. Kolb also interprets the four modi CE, AC, AE, RO as the phases of a cyclic, continuous process as it is shown on the left in Figure 4.2. This view is based on the idea that modes are process steps which entail one another. For instance, concrete experience can be viewed as a tool for reflection, reflection in turn brings forth a draft concept. Such a

concept can again be actively analysed. The learning cycle starts over again as this analysis has brought up new experience.

Studies have shown that – in order to achieve a high knowledge transfer and retention – it is advisable to combine as many as possible of the four modes. The retention rate may increase from 20% to 90% if all four stages instead of a single one are used ([223] cited by [59]). The security lab approach of Crowley [59] is designed according to these findings. It aims at creating security awareness for threats (e.g. by sniffers or scanners) in a computer network. Crowley's training is organized as follows. An introductory lecture with a demonstration (mode: RO) is followed by an exercise where participants are asked to repeat and apply the subject matter (AE) as well as to answer questions concerning the expected outcome of tests (AC). After that, they can play with tools in the lab to concretely experience how network attack and defence works (CE).

Opposed to this, Awareness by Doing puts concrete experience at the first place. More precisely, this means the concrete experience of one's own vulnerability against attacks towards IT systems. After a so-called penetration test (mode: CE) the theoretical background is explained (RO) in order to let participants abstractly understand the dangers (AC). A test lab environment in the demonstration centre offers the possibility for practical experiments (AE). By the combination of all four learning techniques, one can expect a quantifiable improvement compared with other methods that use less techniques. In the following our approach is described in detail.

4.2.4 Organization of the Awareness Campaign

The starting point for a campaign according to the Awareness by Doing concept is the user's everyday working environment. With respect to the phase model of Fox (see Section 4.1.2), our campaign is placed in and extends phases 2 and 3. Before users visit a face-to-face training, they are exposed to simulated security-critical tasks during the normal business day. This part of our concept is called *penetration test at the working place* or *pentest@work* for short. After a presentation of the results gained during the pentest, users receive theoretical and practical training lessons in the demonstration centre in order to build up a healthy measure of mistrust.

Needless to say an Awareness by Doing campaign – like any other awareness campaign – can either be organized and carried out by the company's

own personnel or by external specialists. The latter alternative will probably be, at least for smaller companies, the method of choice as the preparation of the pentests may require less efforts if it is done by an experienced service provider. Besides, particular tools and resources might be necessary, e.g. password cracking tools or servers that host (artificial) phishing web pages.

4.2.4.1 Penetration Test

The pentest is the first and foremost step. The term pentest is deliberately chosen to reflect the similarity to the well-known analysis tool in network or system security (see e.g. [70]). On behalf of the company's management participants are tested and observed during (simulated) security-critical situations at their working place. This phase is strictly limited in time and serves as a means to "pick up people where they stand" with their knowledge and habits. The information collected in this step can be used to illustrate the *status quo* in subsequent training lessons in the demonstration centre.

Obviously, it is necessary to act very sensitively and to absolutely avoid embarrassing single employees. An individual's general right of personality and data protection has to be weighted conscientiously against the learning effect connected with revealing the whole story to the auditorium. If the respective person does not consent, the awareness trainer will in any case only explain the circumstances of the attack and the reaction of the "victim" in a way that is sufficiently anonymized such that the individual cannot be deduced from the information.

Due to legal reasons all measures must be agreed on beforehand with the works council and the responsible for data protection. The design of the pentest should respect the victim's privacy and should avoid to gather personal data and to come into conflict with data protection regulation. For instance this implies that private email must be out of the scope of a simulated attack.

4.2.4.2 Presentation of Results

Rudolph *et al.* [198] emphasize the importance of real examples as a critical success factor in an awareness campaign. Such examples should be directly linked to the respective organisation, the daily work of the employees, and possible consequences of an attack. Concrete details like the attacker's goal and her line of action as well as the sum of possible financial losses or per-

sonal consequences are the best ingredients of a presentation that guarantees immediate impact on the audience. However, these pieces of information are often unknown or not accessible for various reasons. Possibly, attacks have not been detected at all or details could not be revealed due to legal restrictions. The attitude of the management towards the disclosure of past attacks targeting the company may also stand in the way.

This is the point where pentest@work comes into play. It is the method of choice to collect illustrative material for the campaign without doing real harm. The impact on the audience can even be increased when the respective individual under attack consents to an "outing" and describes in her own words how she perceived the attack and which (technical or economic) damages a successful attacker could have caused. Perhaps the victim had in fact received some warning signals, but then had difficulties in interpreting them correctly or transforming them into concrete countermeasures. This experience can be used as the peg to hang on subsequent training, i.e. to convey the symptoms of an ongoing attack as well as to guide people to correct behaviour in such a situation.

4.2.4.3 Demonstration Centre

A demonstration centre offers the unique possibility to let participants reproduce and re-play what was learned in the first step – instead of only *presenting* them the subject matter. Interacting with the system under attack, a user can experiment with various security measures herself and experience their (in-)effectiveness. Plus, she can try to identify concrete symptoms that indicate an attack.

In the lab environment a concrete threat can be demonstrated by means of a memorable example, similar to a "live hacking". This is a good learning opportunity for participants that were not yet included in the pentest@work. Recognizing one's own vulnerability leaves behind a lasting impression at the user. The "aha-experience" is the essential impulse to learn about protection mechanisms and their application. Practical demonstrations go hand in hand with the imparting of the necessary theoretical background. After that appropriate countermeasures are introduced and their use is practised in the lab environment. In such a "sandbox" people can play to their heart's content without being afraid of causing security breaches or doing harm. During that time an expert behind the scenes (similar to the "Wizard of

Oz" in usability testing) tries to compromise the participants' systems via the respective attack vector. The playful character of the whole setting additionally motivates the participants to increase their level of security. Similar methods have been already applied successfully in the context of the evaluation of mail user agents for secure email [253], the so-called "Hacker Contest" at Technische Universität Darmstadt [208], or the UCSB (University of California at Santa Barbara) *Capture the Flag* challenge[8] to name but a few. Small quizzes or competitions can also prove useful in order to motivate a subject and to point out its general difficulties. Consider for instance MailFrontier's *Phishing IQ Test*[9]. In this test participants are shown screenshots of a number of web sites. Some of these are genuine sites of popular services like eBay or Amazon while others are fakes set up by phishers. Past tests have shown that the rate of correct answers is amazingly low.

4.2.5 Success Control

What we propose is a security awareness training method that is completely new in large parts. As such it should be subject to an empirical evaluation in order to answer the question whether the long-term learning results are better than with conventional methods. The realization of a concrete Awareness by Doing campaign and the corresponding evaluation was out of the scope of this thesis due the time frame of several years, which would have been necessary. This is clearly an area for further research.

The evaluation method we consider reasonable is a comparative longitudinal section study in the form of a so-called *panel study*. During such a panel study the set of participants remains identical over the whole coverage period which allows to obtain very precise and individual statements about learning progress and the change in behaviour. Possible tools include the observation of, for instance, the action sequences in the demonstration centre or at the work place as well as successive and periodical interviews. The observation can be automated to some extent depending on the particular type of the attack, e.g. it could be assessed automatically whether people fall for phishing by setting up an appropriate web site and evaluating the log files. Other types of attack may require more manual interaction. We estimate the total length of the study to be around 2–3 years. In this case,

[8] `http://www.cs.uscb.edu/~vigna/CTF/`

[9] `http://survey.mailfrontier.com/survey/quiztest.html`

observations should take place at least every 6 months. On the other hand repeated penetration tests are a means to quantify the lastingness of the training efforts.

4.3 Examples

To give the reader a conception of our approach, we list some single training modules that are conceivable for an enterprise-wide Awareness by Doing campaign. For two modules we will go in a bit more details to give clues how the pentest@work could be like in practice.

The modules can be combined and extended as required. In particular, extensions will be necessary where user groups other than the "average" IT user are to be sensitized (e.g. administrators that are interested in computer forensics or executive personnel that is about to decide about a budget for an awareness campaign). Important modules include, but are not limited to Internet Security, Password Security, Malware, Magnetic Strip and Chip Cards, Physically Accessible Computers, and Wireless Networking.

4.3.1 Internet Security

The purpose of the module *Internet Security* is to teach the most important threats dealing with Internet-based services as we outlined them in Section 2.2. Participants should learn how to react appropriately and get to know reliable sources where they can find information about new threats that affect the majority of Internet users.

Our concept is explained by means of an example for the case of (in)secure email: The test person is an employee of a construction company who is responsible of preparing offers for invitations to bid. During the pentest@work, the user receives an email with a non-authentic sender address requesting confidential information. The alleged sender is an employee of a cooperating company such that the user will probably not be suspicious when asked to reveal confidential data of an offer she is working on. In the preparation of the test, an email address is set up at a popular and well-known ISP which sounds reasonably serious, e.g. AOL or T-Online. The local part of the email address should contain a known person's last name and/or the company's name in order to obfuscate the victim. The text of the email should be somehow tailored to the particular situation, but only use information that

an outsider can gather with some effort, e.g. the department's name, the hierarchy of the company and so on. One idea for an attack vector is the following: The email's sender address is forged to match a valid address of the cooperating company. The alleged employee informs his colleague that he will be out of the office for a few days, but will continue working on the offer. As he has no possibility to access his company's email account he informs the victim to use his private email address instead. A few days later he will then ask for the desired information.

Assume that the test person obeys to this request – possibly after one or two "friendly" inquiries that underline the urgency of the issue. Afterwards this short example would be presented in the demonstration centre. Depending on the consent of the victim, this happens in an anonymous way or not. A similar attack is then simulated by means of a phishing attack before theoretical Internet basics would be discussed. The idea of digital signatures is presented as signatures are one way to fight such threats. Participants would be told what digital signatures are, what security goals can and cannot be achieved with them, and how digital signatures could be put into practice. Depending on the particular needs of the company, the discussion can be broadened to cover topics like authentication on the WWW using SSL (and all its pitfalls we have seen in the two preceding chapters) or the use of data encryption (for emails and/or files). Finally, the test persons would be confronted with similar attacks at a later date. Observation and success control may also include automatic checks on mailboxes in order to measure the amount of digitally signed and encrypted email. For this purpose the statistics tool presented in Section 6.6.2 or an extension thereof could be used.

4.3.2 Password Security

Passwords have always been a key issue of IT security as they are a common mechanism for access control. We have already touched upon this subject in Section 2.2.1 and pointed out the known difficulties users have when they are obliged to manage and recall one or two dozens of different secure passwords. The pentest@work adresses the following three phenomena, which are often found in the wild [202, 257].

Password re-use Identical or similar passwords are used for different services – no matter which. Therefore a goal of the awareness campaign

is to teach participants to distinguish between security-critical and uncritical passwords. For instance the credential used to login in the company's PCs should be considered much more sensitive than one for an Internet forum or a blog.

In order to set a trap to users, the pentest@work may consist for instance of an announcement of an expert web page or newsletter that requires a gratis registration where a password has to be chosen.

Low entropy User-chosen passwords have lower entropy than passwords that are machine-generated, i.e. they are relatively easy to guess as they are too short or derived from words found in a dictionary. It has been shown that – even in companies that have an explicit password policy – one third of the users chose weak passwords [257].

This type of password misconduct can be addressed by brute-force cracking attempts on the employees' system login passwords. Here it is the educational goal of the pentest@work to show that automated password cracking is feasible on weak passwords.[10]

Post-it notes Even in cases where users follow the rules of selecting both strong and different password (or where they are forced to do so by predefined random passwords), they run into trouble recalling the passwords later on. Due to the high complexity, passwords get written down on a piece of paper which is placed near the computer.

As the pentest@work must not invade in the privacy of the employees, the trainer should only look out for post-it notes under the keyboard or on the monitor, but not rummage through other person's desks. A better alternative is to interview room-mates if they have noticed their colleague having to look up passwords before login in.

The most illustrative demonstration is surely about how fast user passwords in general can be found. On a modern PC, the number of passwords

[10]The importance of choosing a suitable threat model has already been pointed out in Section 2.3.2. Here we assume a powerful adversary who can even carry out so-called offline attacks on images of passwords under a one-way function (e.g. the Linux system file `/etc/passwd`). This model also covers the password-based protection of cryptographic softtokens in transit or on hard disk and eavesdroppers on the local network that intercept a login process to a web page protected by HTTP Digest Authentication (see Section 7.2.1) .

that password crackers like *John the Ripper*[11] can probe is on the order of 200.000 per second. The company RainbowCrack[12] already has large precomputed tables of hash values and offers password cracking as a service. A simple back-of-the-envelope calculation shows the security of passwords against brute-force attacks in relation to length and alphabet. These values can be communicated slogan-like.

Before users are invited to a central Awareness by Doing kick-off meeting, the organizers try to find out their passwords. In cases where the brute-force attack succeeds, the passwords are printed out and handed over to the respective persons in a sealed envelope at the beginning of the meeting. These persons are then asked to open the envelope and confirm that what they have got is really their password. After that, requirements towards good passwords are formulated and algorithms are explained that allow users to generate secure, but still usable passwords derived from a secret mnemonic sentence (see e.g. [257]). Furthermore, the training would address practical problems like strategies for the management of multiple accounts or mechanisms for setting file access rights properly to avoid password sharing. A subsequent pentest@work may, for instance, also include social engineering attacks and attempts to elicit passwords by telephone or email.

4.3.3 Malware

This module treats classical instances of malicious code like viruses, Trojan horses, and worms by studying known examples of the past and their destructive potential. Then the correct handling of malware is explained and practised. For this purpose, a sandbox-environment is used where malware can reproduce itself without doing real harm. Malware that spreads fast and produces a drastic reaction is most suitable for educational purposes. If necessary, special demo software should be used. Tools for system or network analysis allow the participants to monitor how fast the malware spreads and to quantify the slowdown.

4.3.4 Magnetic Strip and Chip Cards

The vast majority of the German population possesses at least one chip card or plastic card with a magnetic strip. In this module it is shown how easily a

[11] `http://www.openwall.com/john/`
[12] `http://www.rainbowcrack.com`

magnetic strip can be read out and copied to another card. Test participants are exposed to situations where the card is read with an unauthorized reader, for instance using a counterfeit card reader which asks people for their card in order to grant them access to the canteen. Alternatively, the cashier takes the card away in another room that cannot be observed by the participant. As a second step, it is demonstrated how attackers manage to learn one's PIN, e.g. by manipulated card readers, small video cameras, or shoulder-surfing [192]. In the demonstration centre practical examples are dealt with. Test persons should learn how to recognize genuine card readers and whether a genuine reader (e.g. an ATM) has been tampered with. They should also learn at which occasions it is necessary to enter a PIN.

4.3.5 Physically Accessible Computers

Computers that are physically accessible by an attacker are much more easier to manipulate than systems that can only be reached via a network connection. Users should understand that each of their files can be read out if an attacker sits in front of their machine. During the pentest@work, we try to boot the test persons' computer from a CD or a USB stick and steal a file (e.g. a word processor file that has been recently changed by the user). This file is then presented to the user in the demonstration centre (without revealing the contents to others). Encrypting file systems are discussed as one possible countermeasure against this type of attack.

4.3.6 Wireless Networking

There are two main threats to wireless networks, namely unauthorized access and eavesdropping on the communication between access point and the mobile computer. As the majority of employees is typically not responsible for WLAN access control in their professional context, we concentrate ourselves to eavesdropping during the penetration test. A selected part of the intercepted data is given to the participant and possibly shown in the demonstration centre. Companies often require a user login via a web page before access to the wireless network is granted. Therefore, another idea would be to set up rogue access point and fake the login page (again this targets how users deal with SSL as a security mechanism).

More and more people are also using WLANs at home. For those users we explain standard attacks to gain unauthorized access and show how to

fight them. We present solutions like an adequate encryption and an effective access control mechanism at the access point (using cryptographic techniques).

4.4 Conclusions

In this chapter we introduced a new approach to user sensitisation and IT security awareness training. User studies have shown that users do not believe they are personally at risk and therefore they are not motivated to learn how to operate secure applications properly. This is the reason why our method Awareness by Doing strongly focusses on exposing users directly to threats in order to draw attention, instead of only explaining them what could theoretically happen. By putting motivation at the first place and mixing several learning methods we aim at initiating an effective and lasting change in behaviour. Previous research results suggest that a combination of both should indeed yield significantly better retention rates and a higher lastingness compared to other awareness campaigns.

We sketched the concept of a panel study, which may serve as an evaluation tool to measure the quality of the new method in comparison to other approaches in the long term. As a future project a campaign according to this new paradigm should be planned, carried out, and assessed over the whole duration. We estimate that 2–3 years should be an ideal length for the a campaign to gather reasonable data. The initial costs for preparing the pentest@work units, instructing trainers, setting up the demonstration centre etc. are likely to exceed those of a “classical” awareness campaign. However, we expect that this investment will pay back in the medium term.

Chapter 5

Outsourcing Security to an Organization

When speaking of computer systems,
never use the word "secure".

– Donald Rumsfeld

Trust is good, control is better.

– Vladimir I. Lenin

The discussion and the examples in Chapter 3 give an impression on how challenging it is to set up and operate a trustcenter. And, note that up to now we have mainly looked at the pure technical, PKI-related issues, e.g. the definition of certificate profiles or the realization of the enrolment. But "conventional" safety and security measures are the other side of the coin – a myriad of things falls into this category: for instance physical protection for the trustcenter facility, organizational and personnel security controls, (legal) requirements for audit logging and records archival, general computer and network security issues, or redundancy and fail safe mechanisms (see e.g. [8, 55, 236] to get an impression).

In order to achieve a reasonable level of security in the PKI, obviously one has to take care of a lot of aspects (weakest link property). Companies considering PKI adoption may have the possibility of choosing between "built or buy" provided they have enough resources for their own, in-house trustcenter. For the majority of organizations, and especially for smaller companies,

an outsourced PKI (or "managed PKI" as it is sometimes called) is the alternative of choice. However, although many companies regard IT outsourcing as a good thing, they have reservations about outsourcing security-critical tasks [120]. One important reason for this is the fact that companies do not want to give away complete control over their security [249]. This sceptical attitude proved appropriate as a concrete incident showed, which caused an immense public stir [164, 62]. In this case the service provider had issued at least two non-legitimate code signing certificates on behalf of the customer, putting potentially millions of relying parties at risk of being attacked by malicious code that appears to come from a trusted source.

In this chapter we first look at how outsourcing can be realized in a PKI and what the security implications are. A new method for the effective protection of the enrolment process is presented and analysed in Section 5.2. It enforces a cryptographically strong four-eyes principle, which can be applied both to outsourced and non-outsourced PKIs. A prototype for the issuance of X.509-compliant certificates using RSA or discrete log-based signature schemes (DSA, EC-DSA) is also described. The process is completely transparent for PKI users and relying parties. For instance, it can be retro-fitted seamlessly in an existing trustcenter installation, but one can choose flexibly from a set of integration variants. Secure multi-party computations are an important building block of our method. A short introduction into this cryptographic concept is the subject of Section 5.3. To be effective from a security perspective, our outsourcing approach requires a distributed key generation. We had to rule out existing schemes as they are too slow for our particular case. A new and efficient distributed RSA key generation protocol is described in Section 5.4. Finally, we propose two new applications of threshold cryptography in the field of secure email. One of them tackles the problem of delegation (which is in the focus of Chapter 7 for the case of WWW credentials), the other shows how to protect an enterprise email gateway from key theft.

The presentation in this chapter is based on and guided by our research papers [227, 226, 229], the new enrolment scheme is also the subject of the patent application [231].

5.1 Motivation

In this section we describe the outsourcing scenario with a distinction of CA and RA (registration authority) between service provider and customer. This is the common logical and business model of commercial trustcenters like VeriSign, Entrust, and the like. However, there is a striking weakness in the certificate enrolment protocol that has proven to be security critical. At least one inglorious incident is publicly known where two big players (Microsoft and VeriSign) were involved (see below).

5.1.1 PKI Outsourcing Scenario

In Chapter 3 we have already outlined the main components of a trustcenter and their interdependencies. Here we only consider centralized PKIs as this is the natural choice in an enterprise setting where outsourcing is relevant. The three key modules of a trustcenter are the (actual) certification authority, one or more (local or decentralised) registration authorities, and a certificate/CRL directory (DIR). We assume that there is a division of labour between CA and RA in that the CA is the central and trustworthy party that issues certificates while the RA is responsible of registering users prior to this step. The distinction between those entities is motivated by security reasons, too. Having in mind the specific requirements of CA, RA, and DIR, the canonical solution is to outsource the task of operating the CA and the DIR to a service provider and keeping the RAs in-house. This situation is depicted in Figure 5.1. Let us briefly explain this choice:

In order to provide strong protection and a high confidence level, a CA has to enforce security on multiple layers as we mentioned at the beginning of this chapter, ranging from cryptography, organizational precautions (concerning workflows, trustworthy personnel) to physical protection like access control, brick-and-mortar protection against particular attacks (break-in, destruction etc.). To cut a long story short, these mechanisms, when applied correctly are very expensive, very likely too expensive for a small or medium enterprise when compared to the small number of certificates issued.

On the other hand, the public key directory has large demands for round-the-clock availability. PKIs can only be effective if certificates are available implying that the appropriate directory (typically an LDAP or HTTP interface) is available with very low downtimes. Plus, another task of the

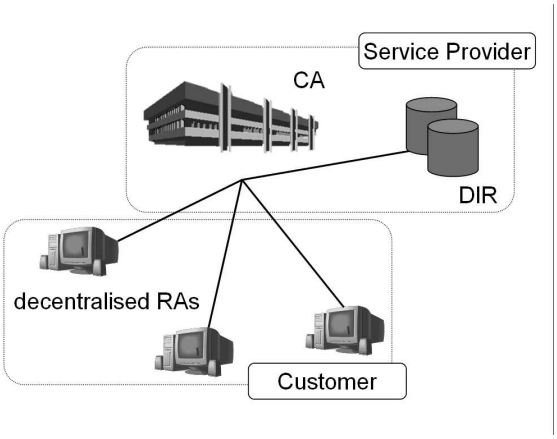

Figure 5.1: General PKI outsourcing setting.

directory is to ensure timely dissemination of revocation lists which is in fact security-critical. Relying clients can be configured to retrieve a CRL and check the certificate in question against this CRL each time a certificate is used. CRLs are short-lived (e.g. 30 days or less) and are thus updated frequently. Experience has shown that directories in fact have to withstand large numbers of requests.[1] Again, the reliability of the directory services is mainly determined by the skills of the provider and monetary factors (yielding to high redundancy, fast hardware, fast network connection etc.). The task of maintaining a directory is itself not security-critical as certificates and CRLs are protected against tampering through the signatures they carry.

5.1.2 A Critical Weakness in the Current Realisation

In the following we sketch a typical enrolment process in order to show an important security risk in the PKI outsourcing scenario described before.

Decentralised Enrolment The process we look at in more detail is the so-called decentralised enrolment. The term "decentralised" signifies that end users generate their key pair locally, e.g. using a dedicated PKI client or a standard web browser (see Section 3.2.4). The whole process is depicted in Figure 5.2. It is initiated by the key pair generation, which goes hand in hand with the creation of the corresponding *certification request*. This

[1] So large that the servers even may break down completely. This happened with VeriSign Inc. in 2004, see `http://computercops.biz/article4646.html`.

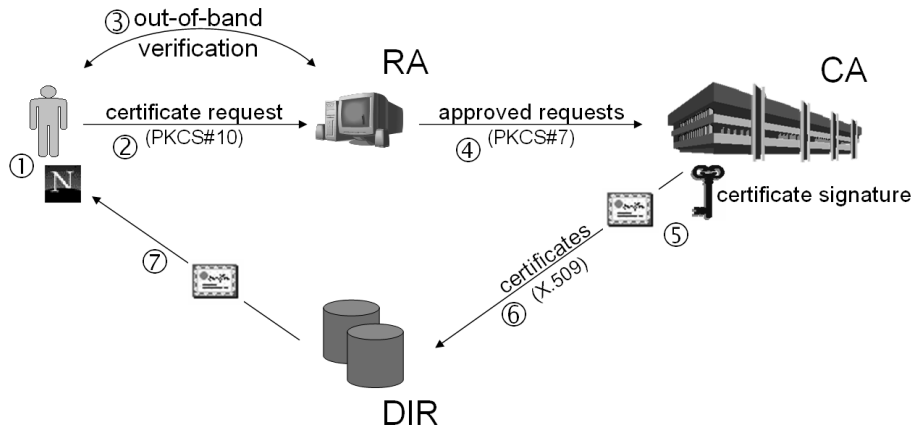

Figure 5.2: Typical certificate enrolment process.

request may, for instance, have the form of a PKCS#10 file (which is the case for browser-generated keys). Alternatively, a user may pack her public key in a self-signed certificate. The request is transferred to the next RA in step 2 where its authenticity (and legitimacy) is checked. This message (step 3) needs to be authenticated, e.g. by the user presenting herself at the RA, to establish a binding between the entity and the public key. In order to trigger the certification process, the RA passes on approved request to the CA in step 4. This may happen in a bulk mode where several requests are bundled in a digitally signed container (like a PCKS#7 file). Note, that this signature has nothing to do with the signature of the certificate. Upon receiving the requests, the CA verifies their integrity and starts the production of certificates. The CA may be offline, i.e. not connected to the network, such that this step may require manual intervention (i.e. carrying a storage medium from one computer to another). Finally, certificates get published in an online directory (step 6). For the sake of simplicity, we assume that end users retrieve their own certificate from DIR (directly or indirectly via a web interface or the like).

Security Critique Security arguments motivate the RA/CA distinction: Firstly, a level of indirection is used to protect the PKI's certification key, its most valuable asset. CAs often run on machines disconnected from the network and far from the end user. The second reason is that a certificate can only be as "strong" (i.e. meaningful) as the underlying registration process.

It is assumed that this process can be best done by decentralised RA offices which are closer to the end user. Behind the sharing of responsibilities there is also the implicit idea to achieve a *four-eyes* or *double verification principle*. This principle claims that two different parties (who might be natural or legal persons) are involved to complete certain crucial processes.

In the PKI setting, the issuance of certificates as the most security-critical task falls into this category. However, in practice the process is not as strong as intended. We claim that there exists a high, immanent risk in an outsourced PKI. As the certificates which the CA issues are not tied to the RA's request, the CA could issue certificates on behalf of the customer in an uncontrollable way. A fraudulent employee of the CA could impersonate legitimate individuals of the company or generate new identities at will – not to mention the theft of the certification key. As a consequence, relying parties may fall for this kind of identity theft. They may place trust in such certificates possibly revealing confidential data to attackers or consider data as authentic which is not.

Attack Vector We believe that this is an intolerable, yet realistic threat. Unfortunately Murphy's Law had proven us right. A severe incident was reported in 2001 when one or more attackers managed to impersonate Microsoft by means of fraudulent certificates [164, 244]. Microsoft uses code-signing certificates issued by VeriSign's CA to disseminate executable code, e.g. Windows updates, in an authentic way. It was detected in spring 2001 that more than two months ago (!) VeriSign had produced two certificates with "OU = Microsoft Corporation, CN = Microsoft Corporation, L = Redmond, S = Washington, C = US, OU = Digital ID Class 3 - Microsoft Software Validation v2, ..." as their subject DN. The certificates were suitable for secure mail and code-signing. As the issuing CA is a pre-configured trust anchor in the Windows OS, the clients would readily access these two certificates as genuine.

The exact circumstances of the incident were not disclosed. It is unknown whether attackers had exploited this weakness and in what form. It is conceivable that Trojan horses may have been distributed with a digital signature declaring them as genuine Microsoft code. The effective window of vulnerability also exceeded the two month between issuance and the publication of the incident and the revocation of the certificates in question:

Although a new CRL was available in VeriSign's directory, clients did not update it automatically as they did not know where to search for CRLs. As a final resort Microsoft had to distribute the CRLs via a software update.

5.2 New Enrolment Protocol

Above we argued that there is an intolerable risk that an outsourced CA may issue certificates that have never been requested. We have seen that indeed this threat is not unrealistic. As a countermeasure we propose a new enrolment scheme. To be effective, a four-eyes principle should be enforced in a cryptographic, not only an organizational way.

In this section we first list four abstract requirements towards a better enrolment protocol. Based on the observations above, these requirements are motivated by security and practical needs. In particular, we demand that the new protocol should be resistant against the before-mentioned attack. Apart from this, it should not make further assumptions and work in the same setting as described above. Based on the requirements we explain the big picture and the details of our realisation from Section 5.2.2 onward. We deal with related work concerning practical applications of threshold cryptography in general in Section 5.2.7. Due to its extent, the cryptographic details of distributed signatures and key generation are postponed to Section 5.3, the body of corresponding related work is reviewed there.

5.2.1 Requirements

For the sake of simplicity, we describe the protocol from the viewpoint of a service provider which serves a single customer. We also assume that there is a single certification authority (denoted CA) which certifies all end entities using the key pair $\langle \mathsf{priv}, \mathsf{pub} \rangle$. Customers may need more than one CAs that issue certificates for different purposes, e.g. certificates for employees who use secure email and certificates for web servers. Extending the model to multiple CAs is straightforward as those CAs are independent from each other in that they have separate key pairs. What we seek is a solution meeting the following requirements:

(R1) The protocol must support multiple registration authorities $\mathsf{RA}_i, 1 \leq i \leq n$, assigned to CA.

(R2) Every creation of a certificate signed by priv requires the cooperation of CA and RA_i for an arbitrary $i \in \{1, \ldots, n\}$. This condition is also sufficient.

(R3) A trusted third party (i.e. different from $\mathsf{CA}, \mathsf{RA}_i$) must not be involved in parts of the protocol where secret information is handled.

(R4) The protocol must output a standard-compliant certificate, which current PKI-clients can verify using pub as a trust anchor.

Let us briefly discuss the motivation behind each condition: (R1) is a natural condition as the customer may want to decentralise its registration offices (see the discussion above). Our protocol provides support for multiple RAs (see Section 5.2.4). The second requirement is due to the observation that, in PKIs using the standard protocol, CA is always able to issue certificates completely on its own, i.e. without a legitimate certificate request (see above).

An external party being substantially involved in the process interferes with the idea of a four-eye principle. This motivates (R3). Consider a scenario where a company outsources CA to a service provider, but remains in charge of the registration. From the company's viewpoint, there is no difference in having to trust – to a large extent – the service provider or another third party (both is undesirable). To be useful in existing PKI environments, the protocol must support standard techniques (especially concerning the certificate format) to avoid the deployment of additional client software (R4).

5.2.2 Big Picture

From now on we assume that the CA is operated by the service provider and the RA is under the control of the customer. using the terms CA (RA) and service provider (customer) as synonyms, we now show how our design fulfils the requirements above.

Shared Digital Signatures The basic idea of the new approach is to apply a secret sharing of priv and to provide the parties with the corresponding shares, say priv_1 and priv_2 such that (R2) is satisfied. This means that the private key is not kept at a single location, but spread over two (or

more) parties such that no party alone can find out the secret value based on its knowledge. More precisely, a *threshold function sharing scheme* [201] is required since priv must not be reconstructed during signature generation (otherwise, a single party would be able to subsequently create certificates on its own, contradicting (R2)). Appropriate schemes for "shared" or "distributed" signatures are available for standard-compliant RSA [36] and DSA [159]. They take as input the message m to be signed and return $\mathtt{sign}_{\mathsf{priv}}(m)$, the regular digital signature of m with the key priv. These algorithms require multiple local computations of each party and two or four rounds of communication, the details of which are given in Section 5.3.2. In any case, a relying party can verify $\mathtt{sign}_{\mathsf{priv}}(m)$ solely with pub as usual and is ignorant of the way of how the signature was generated.

The Need for Distributed Key Generation During an initialisation phase, a private key and the secret shares for the respective parties have to be generated. We cannot delegate this task to a *trusted dealer*, a party different from customer and service provider. The reason is (R3). The parties have to use a shared key generation protocol instead because the dealer may ally himself with one of the parties in order to discover priv. Fortunately, protocols for shared key generation are available, too. While the shared generation of DSA keys is very efficient for the two-party case, this did not hold so far for RSA. An efficient protocol, which is due to Boneh and Franklin [32], allows three or more parties to generate an RSA key pair in a way such that neither of the parties obtains information about the private key. However, if two of three parties in the Boneh-Franklin protocol work together, they can find out the key. This precludes the involvement of a third entity as a *helper* party because malicious coalitions of customer and helper or service provider and helper again could reconstruct the key. We devised a scheme tailored for the two-party case which is significantly more efficient than previously known methods. It is the subject of Section 5.4.

5.2.3 Protocol Variants

Our new approach requires no fundamental change in the processes in a PKI and therefore allows an integration into existing structures. As usually, we assume that the registration takes place in a decentralised fashion as it was the case before. The concrete realization depends on the signature scheme

that is used, the mapping of the roles in the distributed scheme to the CA and RA, and the way of how the corresponding protocol messages are added to the process shown in Figure 5.2. As stated above, a distributed RSA or DSA signature requires a 2-way or 4-way communication respectively. We assume that the message to be signed, i.e. the certificate body, is a priori known to both parties. We assume that priv_1 and priv_2 denote shares of a private key which can be reconstructed by combining them. This relation is expressed by writing $\mathsf{priv} = \texttt{combineshares}(\mathsf{priv}_1, \mathsf{priv}_2)$.

Full Integration Supporting Multi-Way Protocols The most generic way to integrate the distributed signature into the enrolment protocol is to modify step 5 of the protocol shown in Figure 5.2. As CA does no longer create the signature on its own, a bilateral communication with the RA is necessary. We call this variant "full integration" as it requires certain changes on both sides. It is depicted in Figure 5.3. A benefit of this approach is its generic nature as it can be used with any distributed signature scheme, no matter how many communication rounds it requires. Besides, the roles in the distributed signing protocol can be assigned to the CA and RA in an arbitrary way.

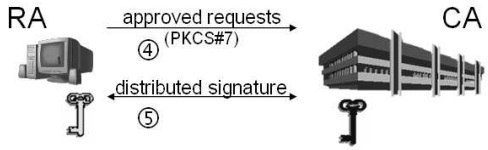

Figure 5.3: New enrolment protocol – full integration variant.

However, this variant creates the need either for an online CA, which is able to communicate with the RA in real time, or for manual processing as the data has to be transported from a machine connected to the network on to the offline CA and back. Offline CAs always require such manual data transport, but the distributed signing protocol increases the number of such steps. In order to minimize the changes of interfaces at the side of an online CA, the functionality for shared signatures could be encapsulated in a PKCS#11-compatible "proxy" that appears as a black box to the trustcenter software at the CA. Such a proxy would open a network connection to an RA process and would access CA's local key store via PKCS#11 [194] (or

Figure 5.4: New enrolment protocol – request-less certification variant.

another standard interface). This allows CA to re-use existing HSMs or SmartCards.

Request-less Certification The name of this variant is inspired by the observation that the approved requests in step 4 in Figure 5.2 could be omitted in favour of "partially" signed certificates in case a 2-way communication scheme for the signature is used. We consider an abstract 2-way scheme, which we modelled by two functions representing each of the protocol messages: Let `partialsign` denote a place holder for the computation of the first party in the protocol, it takes the message to be signed m and the private key share priv_1 as input. The value $\sigma := \mathtt{partialsign}_{\mathsf{priv}_1}(m)$ is sent along the channel. The second applies the function `countersign` to its private key share priv_2 and the value σ to obtain

$$\mathtt{countersign}_{\mathsf{priv}_2}(\mathtt{partialsign}_{\mathsf{priv}_1}(m)) = \mathtt{sign}_{\mathsf{priv}}(m).$$

If RA is assigned the first role and CA the second, one obtains the variant depicted in Figure 5.4. In comparison to the previous variant, it requires one communication step less under the assumption that RA already knows the complete contents of the certificate body. As a result of the registration process, an RA always knows the public key and the name of the subject to be certified and the name of the corresponding CA. The remaining variable parameters are the certificate serial number and its validity period. The former may be either obtained by increasing an internal counter, which is known by RA and CA or by taking the hash value of the subject public key, which is also common practice [128]. The latter may be determined by the RA and cross-checked by the CA.

Seamless CA Integration with Post-Processing We can obtain another variant for the 2-way case by exchanging the roles of the parties in the

shared signature protocol. Now CA computes `partialsign` and RA countersigns the intermediate result. Under the condition that for all m and priv_1 the equality

$$\mathtt{partialsign}_{\mathsf{priv}_1}(m) = \mathtt{sign}_{\mathsf{priv}_1}(m)$$

holds, this variant allows seamless integration in an existing trustcenter at CA. The only thing that has to be changed there is the prior generation of a key pair $\langle \mathsf{priv}, \mathsf{pub} \rangle$ (with an extra tool) and step 5. Here we let CA use a share priv_1 of the private key instead of a "complete" private key. The condition above holds for RSA such that the post-processing variant should be adoptable by any trustcenter software as RSA is the most common signature scheme. Obviously, the output of CA in this case is an X.509 certificate without a valid signature with respect to the public key pub. As such it must not be published immediately on the directory.

This is where the post-processing is brought up. One can think of two different directories, one that is publicly accessible and one that temporarily stores the pending certificates coming from the CA. An alternative is to use only one directory and write the pending certificate of a user in a hidden entry which is only visible by the RA. In each case a process at the RA scans the directory for new entries, checks and countersigns them, and writes the complete certificates back at the appropriate location. This variant is depicted in Figure 5.5.

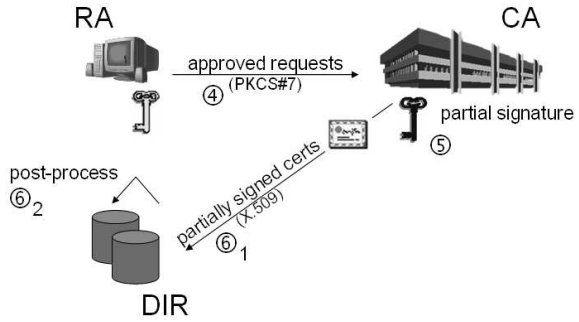

Figure 5.5: New enrolment protocol – post-processing variant.

5.2.4 Multiple RAs

The first of our requirements was the support of multiple RAs. Our approach in fact allows multiple RAs, but does not mandate that all of them

use the same enrolment variant. There are basically two ways which differ in the key pair they use. A straight-forward variant is to use a two-stage hierarchy where service provider and customer share the private keys $\mathsf{priv}^{(i)}, i = 0, 1, \ldots$ of several key pairs, say $\langle \mathsf{priv}^{(0)}, \mathsf{pub}^{(0)} \rangle, \langle \mathsf{priv}^{(1)}, \mathsf{pub}^{(1)} \rangle, \ldots$ The key pair $\langle \mathsf{priv}^{(0)}, \mathsf{pub}^{(0)} \rangle$ has a special role as it is used to issue certificates for the other public keys $\mathsf{pub}^{(1)}, \mathsf{pub}^{(2)}, \ldots$ The situation is depicted in Figure 5.6.

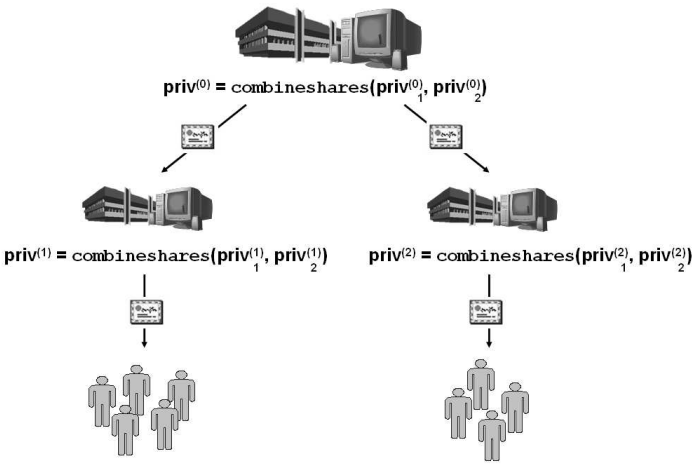

Figure 5.6: Multiple RAs – hierarchical variant.

In this variant new RAs can be added easily by letting service provider and customer generate a new key pair in a distributed protocol and have its public key certified by the topmost $\mathsf{priv}^{(0)}$. A small drawback of this approach is the fact that the certificate chain gets longer and the chains differ for certificate holders of the same company. This should usually be no problem. If it is, it can be avoided by the following approach: Only one key pair $\langle \mathsf{priv}, \mathsf{pub} \rangle$ is used, but this key pair is shared in several different ways, i.e. we have

$$\begin{aligned} \mathsf{priv} &= \mathtt{combineshares}(\mathsf{priv}_1^{(1)}, \mathsf{priv}_2^{(1)}) \\ &= \mathtt{combineshares}(\mathsf{priv}_1^{(2)}, \mathsf{priv}_2^{(2)}) = \ldots \end{aligned}$$

In this case customer and service provider set up the system by generating one key pair (in a distributed protocol). Later they can add RAs as required by computing new shares. This process is called the *re-sharing* of priv. For all i the shares $\mathsf{priv}_1^{(i)}$ remain with the service provider while

the $\mathsf{priv}_2^{(i)}$ are known by the decentralised RAs. Depending on which RA processes a request, the CA selects the appropriate share $\mathsf{priv}_1^{(i)}$. The resulting certificates do not reveal which RA was actually involved. This situation is depicted in Figure 5.7.

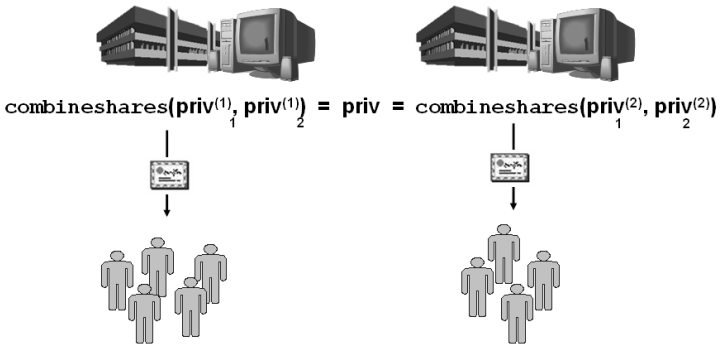

Figure 5.7: Multiple RAs – single key pair variant.

5.2.5 Security Properties

All of the three proposed protocol variants solve the problem of enforcing a cryptographically four-eyes principle between RA and CA. This method is by far stronger than relying on organizational and procedural controls or some sort of RA/CA trust relationship. Only cryptography can effectively prevent the CA from deviating from the enrolment protocol. It can be shown that neither the service provider, nor the customer has enough information to issue certificates on its own. This also holds for the case of multiple RAs. Exemplary proof details are given in Section 5.3.

One might argue that other mechanisms could also do to prevent fraud in an outsourcing scenario. For instance, the service provider could commit to a legally binding CP/CPS. Furthermore, the service provider could be made the obligation to archive certificate request in a tamper-proof and verifiable way. However, the secure long-term archival of signed data is a complex and costly process [37]. Besides, it is not very effective against the threat of a malicious CA as the customer has to suspect fraud and has to have the corresponding certificate in hand before it can demand the CA to show a valid request.

There is also not the same level of security if one tries to implement a four-eyes principle on the software or operational level. Consider the prac-

tical aspects of a trustcenter operator: there are typically a larger number of roles to be matched to internal processes – for an example see e.g. [236] where the role model of FlexiTrust is described. Among those roles are system administrators who configure and maintain the hard- and software. This is an obvious crunch point in the relation between customer and service provider. At the system management level a four-eyes principle between the two cannot be enforced when the platform is under the sole control of the service provider's personnel.

5.2.6 Prototype

As a proof-of-concept, a prototype for the new enrolment protocol has been implemented. It is written in Java using the FlexiProvider[2] cryptographic library and the Fraunhofer ASN.1 Codec[3]. The package supports all three enrolment variants, RSA as well as DSA, and both variants for multiple RAs. For the latter purpose, there is a `Split` class in the package which takes a key pair, computes shares of the private key, and stores them in the Java keystore or PKCS#12 format. This class can also be used to simulate key generation for testing. (Distributed RSA key generation is a time-consuming task – even with our optimized algorithm, as we can see in Section 5.4.) The various shared signature generation methods take as input a self-signed (end entity) certificate instead of a PKCS# 10 request. For testing purposes such certificates can easily be generated by using e.g. OpenSSL or Java's keytool. The content of the certificate with a shared signature can be configured freely.

RSA The classes `Sign` and `Countersign` provide methods to locally compute partial RSA signatures[4] and realize the combination of two partial signatures into a single one respectively. Similar methods are available in a network-enabled version that can be used in the first and the second enrolment variant. The real-time online protocol is realized by TCP/IP socket communication. As a consequence all payload data can be secured by tunnelling the communication through SSL or SSH.

[2] `http://www.flexiprovider.de`

[3] `http://www.semoa.org`

[4] Remember that $\texttt{partialsign}_{\mathsf{priv}_1}(m) = \texttt{sign}_{\mathsf{priv}_1}(m)$ holds in the case of RSA.

The computational costs for `Sign` and `Countersign` are nearly identical as the latter differs from the former only by one modular multiplication. The costs for each party are higher by a factor of 4 compared with the standard RSA signature computed by one party alone. By way of experiments we found that the signing (and hashing) of a 1 KByte message with a 2048 bit key takes 0.636 seconds instead of 0.171 (Pentium IV processor with 1.8 GHz, averaged over 100 measurements). The reason for this slowdown is the fact that the parties cannot use the Chinese remainder theorem (CRT) to speed up computations as they do not know the factorization of the modulus. Up to now, we considered standard RSA key pairs with a two-prime modulus. In case our new shared key generation scheme is used, the outcome is a three-prime modulus. The average time in this case is 0.240 seconds (here CRT can be applied as each party knows two of the three factors).

DSA As the DSA shared signature scheme of MacKenzie and Reiter [159], which we have chosen to implement, is a 4-move protocol, only the generic enrolment variant can be supported. The time for shared signature generation over a TCP/IP link is 2.097 seconds on average. Here we used a 1024 bit key ($\log_2(p) = 1024, \log_2(q) = 160$).

In the current prototype we made use of the Paillier cryptosystem [180] as a building block in the protocol. Other semantically secure cryptosystems with a similar homomorphic property are conceivable (see Section 5.4.1.1 for further examples). The modular structure of the code allows to exchange them flexibly. This is a desirable feature as one often does not want to rely on a single security assumption.

Furthermore, our package contains routines for the shared generation of DSA key pairs (also in the MacKenzie-Reiter setting). An experimental implementation shows that shared key generation over the network takes less than one minute (43.1 seconds) on average for 1024 bit keys.

5.2.7 Related Work

Threshold cryptography is already used in several applications: Examples include a "virtual SmartCard" realized by splitting a private key into two halves, one derived from a user password and the other one stored on a central server (see [158] and `www.singlesignon.net`). Such a virtual SmartCard allows instant key revocation similar to the online semi-trusted party

which uses "mediated RSA" described in [31]. It has been shown in the course of the ITTC project[5] how the private key of an SSL-enabled Apache web server can be protected with threshold cryptography. Threshold cryptography was also a design criterion for the protection of the signing key in SET (Secure Electronic Transaction, see `hhtp://www.setco.org`).[6] Distributed RSA signing was also used in the context of a time stamping service hosted on three or five serves [13]. During setup, the servers engage in a distributed protocol to generate a shared RSA key.

A four-eyes principle between RA and CA could also be enforced in a related, but slightly different way. The same security level is achieved if all certificates that are issued carry two digital signatures from independent parties. A similar idea can be found in fail-safe concepts (cf. page 39) where two mathematically independent algorithms are used. Here the purpose is to face the risk that a signature algorithm may become cryptographically weak over time [161]. Although, double signatures are not conforming to X.509 (and would violate the requirement (R4)), they are explicitly an option in the OpenPGP certificate format [48]. Nevertheless, there are some drawbacks to this ad-hoc solution: Firstly, the costs for certificate validation are doubled as two signatures have to be checked. Secondly, such a policy saying that both signatures should be verified cannot be enforced with current applications. Thirdly, double signatures reveal internal details of the company's PKI, which may not be desirable for various reasons.

5.3 Secure Multi-Party Computations

In this section the problem of distributed key generation is addressed. The two-party setting is the most important one for our application. Nevertheless, we give the reader a short introduction and a survey on related work by looking at the general case of multi-party computations and multi-party key generation respectively.

We postpone the treatment of discrete log-based signature schemes to Section 5.3.2 and now look at RSA, which is without doubt still the most wide-spread digital signature algorithm in use today. There are in fact ef-

[5] Intrusion Tolerance via Threshold Cryptography, see [256].

[6] Note that SET also uses a mechanism called *dual signatures* [70]. This must not be confused with threshold signatures as here the word dual refers to the simultaneous signature (with a "standard" private key) of *two messages*.

ficient algorithms to generate an RSA key pair in a distributed fashion, namely the protocol of Boneh and Franklin which has been shown to be efficient for practical purposes. However, this protocol cannot be transferred literally to the case of less than three parties which excludes it from our scenario. Attempts have been made to solve the problem of two-party RSA key generation. Proposals are due to Cocks [56, 57], Gilboa [102], and Poupard/Stern [185]. A first prototypical implementation of Gilboa's protocol and two of the proposed variants (homomorphic encryption and oblivious transfer) therein [140] suggests that this algorithm is not very practical. The time for key generation on a local area network (LAN) is more than three hours for a 1024-bit RSA key using the homomorphic encryption variant. This would be by far too slow for most applications.

We thus propose a new protocol that is asymptotically faster than all previous two-party schemes. Those schemes have in common their quadratic complexity (in the bit length of the modulus). Our scheme removes this structural weakness as it has only linear complexity. Performance measurements have shown that key generation with our algorithm takes on the order of half an hour for a 1536-bit RSA key. Since our methodology falls back on techniques from several papers ([32, 34, 25, 102]), these ideas are summarized in the following paragraphs.

5.3.1 General Setting

Before considering distributed key generation algorithms in particular, we briefly recall the basics of general secure multi-party computations. We do not go too much into details here but refer the reader to the literature for a thorough treatment (e.g. [49, 108]). In the following, we mainly adopt the terminology and notation of Goldreich's textbooks [107, 108].

5.3.1.1 Network and Attacker Model

In a typical setting for a distributed multi-party protocol, a fixed number, say m, of parties is pairwise connected by a (reliable, but not necessarily private) communication channel. Each party contributes a secret local input s_i and – in order to allow randomized algorithms – the content r_i of its "random tape" to the protocol. The parties exchange messages with each other in order to execute a distributed "software application" that computes a function F mapping an m-ary input vector $\bar{s} := (s_1, \ldots, s_m)$ to an output

vector of the same length. For the sake of simplicity, it is assumed that the message exchange is organized in a number of *rounds* such that in each round, each party sends (and receives) exactly $m - 1$ messages (some of them may be the empty string λ). After the last round, each party obtains its own result (or "local output") which equals the respective coordinate in the result vector $F(\bar{s})$.

Adversaries In this setting, an adversary may corrupt a subset of the parties that take part in the computation. Such a subset is often called a *coalition* while the respective parties are said to be "dishonest" or "corrupt". Loosely speaking, the protocol should provide security in that the dishonest parties do not obtain information that is not already implicitly given by their local inputs or the result of the computation. The idea behind this approach is logically based on the so-called "simulation paradigm" where one thinks of a practical protocol as an "emulation" of its "ideal" counterpart. Such an ideal protocol relies on a trusted third party that runs the desired algorithm, which is the same as in the real counterpart. It has a rather simple communication model as the parties just submit their inputs to the trusted party which does all the computations alone and then in turns sends back the outputs, one to each party. The (real) protocol is considered secure if the advantage of an adversary is the same as in the imaginary model. Precise definitions are given below.

Adversaries can be characterized by

- their *computational power* (e.g. probabilistic polynomial time or unbound),
- their *behaviour* during the execution of the protocol ("active" or "passive", see below),
- the (maximum) *number* (or ratio) of corrupt parties, and
- the information whether the set of dishonest parties is *fixed* for the course of the protocol.

Throughout this section we consider coalitions that are arbitrary, but fixed before the protocol starts. We restrict ourselves to adversaries that behave passively. Such adversaries are also called "semi-honest" as they follow the protocol to the letter, but are allowed to record all the messages they

receive and do arbitrary extra computations on them. This is also called the *honest-but-curious* scenario. Observe that there is a generic technique [108] to turn a two-party protocol that is secure in this scenario into an equivalent protocol that is secure against active adversaries. Active adversaries are more powerful since they are not obliged to adhere to the specified protocol.

5.3.1.2 Notion of Security

In order to be able to extend the model to randomized protocols, we use the idea of *functionalities* as a coarsening of the concept of a function. An m-ary functionality F denotes a random process that maps a vector of m inputs to a vector of m outputs. Functionalities can be thought of a probability distribution over a set of corresponding m-ary functions $f^{(i)}$ such that F equals $f^{(i)}$ with a certain probability p_i.

Indistinguishability of Probability Ensembles In the following we have to deal with probability ensembles. A *probability ensemble* is a family of random variables X_n indexed by a countable set (typically the set of strings or the set of positive integers). Security is described in terms of indistinguishablility of probability ensembles. Two ensembles $X := \{X_n\}_{n\in\mathbb{N}}$ and $Y := \{Y_n\}_{n\in\mathbb{N}}$ are said to be *computationally indistinguishable* if for every non-uniform family of polynomial-size circuits $\{C_n\}_{n\in\mathbb{N}}$ (the "'distinguishing algorithm"') the difference $|\text{prob}[C_n(X_n) = 1] - \text{prob}[C_n(Y_n) = 1]|$ is negligible in n. Here, a function $f(n)$ is called *negligible* in n if for every positive polynomial $p(n)$ (i.e. $p(n) > 0\ \forall n \in \mathbb{N}$) there exists an n_0 such that $f(n) < 1/p(n)$ for all $n > n_0$.

A stronger notion, which does not refer to a possible distinguishing algorithm, is that of *statistical indistinguishability.* The ensembles are called statistically indistinguishable if $\sum_\alpha |\text{prob}[X_n = \alpha] - \text{prob}[Y_n = \alpha]|$ is negligible in n. If the sum vanishes, we say that the ensembles are *identically distributed* and speak of *perfect indistinguishability.* We denote computational and statistical indistinguishability by $X \stackrel{\text{c}}{\equiv} Y$ and $X \stackrel{\text{s}}{\equiv} Y$ respectively.

Security for Distributed Protocols We can now give a formal definition of a secure multi-party protocol Π. We use the notion of a party's *view* to describe the values a party gets to know during the protocol with input $\bar{s}$. The view of party P_i is a tuple $\text{View}_i^\Pi(\bar{s}) := (s_i, r_i, m_1^{(i)}, \ldots, m_l^{(i)})$ where

s_i is again the party's input, r_i represents the outcome of its internal coin tosses, and the $m_j^{(i)}$ are the messages received in round j of the protocol. (When it becomes clear from the context which protocol we are talking of, the parameter Π is ommitted.) The joint view of a coalition that consists of the parties $\{P_i : i \in I\}, I \subseteq \{1, \ldots, n\}$ is the tuple $\mathrm{View}_I^{\Pi}(\bar{s})$ that contains the index set I and $\mathrm{View}_i^{\Pi}(\bar{s})$ for each $i \in I$.

Definition 3 (Private Computations) *Let f be a functionality. The protocol Π is said to t-*privately compute *compute f if there exists a probabilistic polynomial-time algorithm S (a "simulator") such that for any coalition $\{P_i : i \in I\}$ of at most t parties the following holds*

$$\{[S(I, \bar{s}_I, f_I(\bar{s})), f(\bar{s})]\}_{\bar{s}} \stackrel{c}{\equiv} \{[\mathrm{View}_I^{\Pi}(\bar{s}), \mathrm{Output}^{\Pi}(\bar{s})]\}_{\bar{s}}.$$

Here, $\bar{s}_I$ and f_I denote the projections on the coordinates in I and $\mathrm{Output}^{\Pi}(\bar{s})$ denotes the vector that contains the output for each party after the protocol was run on input $\bar{s}$. If the two ensembles are even *identically distributed*, the simulation is called a *perfect emulation*. More specifically for a fixed I, we say that the protocol preserves privacy against the particular coalition I or party if I has only one element. We also use the adjective I-private. In a two-party setting we call the computation private or say that it preserves privacy if it is 1-private.

5.3.2 Threshold RSA and DSA

Let us briefly sketch how the distributed signing of certificates works. Again m denotes the certificate body to be signed respectively its hash value plus an appropriate padding (e.g. RSA-PSS, see [197]). Observe that the input for the signature scheme can be prepared by one party alone, the second party could check and reject the signature should the occasion arise. We only describe threshold signatures for RSA in detail here as the protocol for DSA is a bit lengthy, the interested reader is referred to [159] for details. We sketch how distributed DSA key generation works and explain the protocol for re-sharing of the private key

RSA The RSA encryption and signature scheme is standardized by the document PKCS#1 [197]. Multi-prime RSA, which was formerly defined in an amendment, is now incorporated in version 2.1 of the standard. A valid

RSA modulus $N := \prod_{i=1}^{u} r_i, u \geq 2$, is a product of distinct odd primes r_i. In case $u = 2$ the system is called (standard) RSA and otherwise *multi-prime RSA*. Two integers $0 < e, d < \varphi(N)$ satisfying $ed \equiv 1 \mod \varphi(N)$ are needed. They are called public and private exponent respectively as $\mathsf{pub} = (N, e)$ and $\mathsf{priv} = d$. A value m is signed by computing $m^d \mod N$, the verification process computes the e-power to obtain m back again.

For the purpose of threshold cryptography, the private exponent has to be shared among the parties. A straight-forward way to do so is an *additive sharing* $\mathsf{priv} \equiv \mathsf{priv}_1 + \mathsf{priv}_2 \mod \varphi(N)$ [36]. In this case, the functions `partialsign` and `countersign` are defined as follows.

$$\texttt{partialsign}_{\mathsf{priv}_i}(m) := m^{\mathsf{priv}_i} \mod N,$$
$$\texttt{countersign}_{\mathsf{priv}_i}(\sigma) := m^{\mathsf{priv}_i} \cdot \sigma \mod N,$$

Using the notion and notation of the previous section, it is easily seen that the computation is private: Assume that party i holds priv_i, party 1 computes `partialsign` while `countersign` is executed by party 2. We have $\bar{s} = (\mathsf{priv}_1, \mathsf{priv}_2), F(\bar{s}) = (\sigma_2, \sigma_2)$ where $\sigma_2 = \texttt{sign}_{\texttt{combineshares}(\mathsf{priv}_1, \mathsf{priv}_2)}(m)$ is the local output of each party. As RSA signing is a function (and not a functionality), the random tapes are empty: $r_1 = r_2 = \lambda$. The distributed signing protocol has only two rounds, the protocol messages are $m_1^{(1)} = m_2^{(2)} = \lambda$ and $m_1^{(2)} = \texttt{partialsign}_{\mathsf{priv}_i}(m) = \sigma, m_2^{(1)} = \texttt{countersign}_{\mathsf{priv}_i}(\sigma) = \sigma_2$. The simulation of

$$\text{View}_1(\bar{s}) = (\mathsf{priv}_1, \lambda, \lambda, \texttt{countersign}_{\mathsf{priv}_2}(\sigma))$$

is trivial as the simulator gets priv_1 and σ_2 as inputs. The simulation of

$$\text{View}_2(\bar{s}) = (\mathsf{priv}_2, \lambda, \texttt{partialsign}_{\mathsf{priv}_1}(\sigma), \lambda)$$

is done by computing $\sigma_2 \cdot m^{-\mathsf{priv}_2} = m^{\mathsf{priv}-\mathsf{priv}_2} = m^{\mathsf{priv}_1}$. We have therefore shown the following proposition. (A similar statement holds for DSA.)

Proposition 4 (Privacy of distributed RSA Signatures) *The shared computation of RSA signatures preserves privacy.*

DSA A DSA key pair is a tuple $\langle (p, q, g, x), (p, q, g, y = g^x \mod p) \rangle$ where p and q are primes such that q divides $p - 1$, $g \in \mathbb{Z}_p^*$ is of order q. The private key is $\mathsf{priv} = x \in \mathbb{Z}_q^*$ which is chosen uniformly at random.

Initially, CA and RA_1 construct q, p, g in an interactive protocol (in this order): q can be defined as the smallest prime exceeding a random lower bound which was agreed on using distributed coin-tossing (see e.g. [178] how to implement such a protocol). The construction of p and g happens in a similar way. CA and RA_1 select secret random values w_1 and x_1 which determine the private key via a *multiplicative sharing* $x \equiv w_1 x_1 \mod p$. The values $y_1 \equiv_p g^{w_1}$ and $z_1 \equiv g^{x_1} \mod p$ are then published such that both parties can compute $y \equiv y_1^{x_1} \equiv z_1^{w_1} \mod p$.

We now describe how re-sharing, i.e. the introduction of further registration authorities, works. This is done one RA after another. We assume that $\mathsf{RA}_1, \mathsf{CA}$ and the new RA_i are connected by secure channels. $\mathsf{RA}_i, i > 1$, obtains its share x_i by a technique similar to that used in *proactive and verifiable secret sharing* (see e.g. [78, 96] for further information about those topics): RA_1 picks $u_i \in \mathbb{Z}_q^*$ at random and sends $(i, u_i, z_1^{u_i^{-1}})$ to CA and $(x_i = x_1 u_i^{-1} \bmod q, g^{u_i})$ to RA_i. CA and RA_i can verify that the pair $(w_i = w_1 u_i \bmod q,\ x_i)$ is indeed another multiplicative sharing of x by exchanging the values $y_i = g^{w_i} \bmod q$ and $z_i = g^{x_i} \bmod q$. It is not difficult to prove

Proposition 5 *The computation of a DSA key re-sharing is both* $\{\mathsf{RA}_1, \mathsf{RA}_i\}$*-private and 1-private.*

5.3.3 The Protocol of Ben-Or, Goldwasser, and Wigderson

The protocol of Ben-Or, Goldwasser, and Wigderson [24] is a nice example for a $\lfloor \frac{m-1}{2} \rfloor$-private multi-party computation. It is heavily used in the Boneh-Franklin scheme (see Section 5.3.4.1). In the following, we abbreviate this protocol by BGW. BGW allows m parties to jointly evaluate a function F that is realized as an *arithmetic circuit* composed of *gates* for binary addition, multiplication, and for constant addition or multiplication. (See [108] for a similar construction using Boolean circuits and [54] for an overview of general constructions.) BGW makes use of Shamir's well-known secret sharing scheme [211], which we briefly recall before going into details of the BGW protocol in the next but one paragraph.

Shamir Secret Sharing Shamir's scheme is based on the fact that the coefficients of a polynomial over a finite field E with degree less than k can be reconstructed by knowing k different points $(x_i, f(x_i))$ on its curve. To

share a secret $s \in E$ among the *share holders*, an element $f(X) \in s + Q_k$ is chosen uniformly at random where

$$Q_k := \{f_1 X + f_2 X^2 + \cdots + f_{k-1} X^{k-1} : f_1, \ldots, f_{k-1} \in E\}.$$

is the set of polynomials of degree at most $k-1$ and vanisihing free coefficient. s is thus "encoded" as the free coefficient $f(0)$ of the polynomial. Shares have the form $(x_i, y_i := f(x_i))$ where $x_i \neq 0$ is a fixed public value for the i-th party, $1 \leq i \leq m$ (obviously, the cardinality $|E|$ must exceed m, the number of share holders). This is called a k-out-of-m secret sharing. In a typical application of Shamir's scheme, f is chosen by a trusted party, the so-called *dealer*, which distributes the shares during a setup stage and then leaves the protocol. For instance, the software package PGP offers such a secret sharing for the backup of decryption keys. In this case, the dealer is the owner of the private key and the share holders are trustworthy friends, colleagues and the like.

It is easily seen that, given less than k points on the curve of f, each element of E is equally likely for s [41]. Therefore, even a coalition of $k-1$ parties that reveal their shares to each other does not learn anything about s. This statement holds in an information-theoretic sense.

Distributed Private Evaluation of Arithmetic Circuits All BGW computations are done in a finite field E where again $|E| > m$. The function F is realized by an *arithmetic circuit*. Apart from the constants, all intermediate values, i.e. the inputs and outputs of each gate, as well as the result of the whole computation, are distributed according to a k-out-of-m Shamir scheme. BGW is based on the following observation: Assume that a and b are inputs to a gate and that a and b are shared using the polynomials $f \in a + Q_k$ and $g \in b + Q_k$ respectively. Hence $(x_i, f(x_i) + g(x_i))$ are shares of the polynomial $h = f + g$ which determines the sum $a + b$ by its free coefficient. Shares of the values $\kappa + a$ and μa for publicly known constants $\kappa, \mu \in E$ are computed in a similar way.

The first step of a BGW protocol is the *input stage* during which each party makes its input s_i available to the other parties via a k-out-of-m secret sharing. After that the parties enter the *computation stage* where they simulate each gate in a distributed private computation. Each of these computation preserves $(k-1)$-privacy. The last gate in the network produces a sharing of the desired result $r = F(\bar{s})$. All parties broadcast their shares

of r such that each of them is able to interpolate the polynomial that determines r. It is easily seen that the evaluation of an affine linear function $F : E^m \to E$ (which can be computed with the three previously introduced gates) can in fact happen in an $(m-1)$-private way when an m-out-of-m secret sharing is used.

Things are slightly different for the *multiplication* of two values a and b. Again, we assume that $a = f(0)$ and $b = g(0)$ are shared based on the polynomials f and g. In this case, an m out of m secret sharing cannot be used since the degree of the product $f \cdot g$ may exceed $m-1$ and therefore interpolation may not possible any more. One has to use a $(k+1)$-out-of-m sharing where $k \leq \lfloor \frac{m-1}{2} \rfloor$ to ensure that $\deg(f \cdot g) \leq 2k \leq m-1$. There is another, yet more subtle, detail one has to pay attention to: The product of two (random) polynomials is not random any more. This prevents a straightforward computation of $a \cdot b$. As a remedy, one can use the re-randomizing technique outlined in [24] or the extrapolation formula used in [108]. We briefly describe how the former method works.

As the result of the computation should be shared in the same way than the inputs, that is, in an $k+1$ out of m sharing, the first step is to map the polynomial $f \cdot g = \sum_{i=0}^{m-1} c_i X^i$ to one of smaller degree. It can be shown (see [24]) that the function that maps this product to the partial sum $h(X) := \sum_{i=0}^{k-1} c_i X^i$ is basically a single matrix multiplication. As matrix multiplication consists of affine linear functions, this computation is $(m-1)$-private. In particular the matrix is publicly known as it only depends on the $x_i, 1 \leq i \leq m$. Now, the second step consists of randomizing the polynomial h. In order to do so, each party P_i chooses an element $q_i \in Q_k$ at random and distributes the shares $q_i(x_j), 1 \leq j \leq m$, to the respective parties. If at least one of the q_i is random, the coefficients apart from the free term of $h + \sum_{i=1}^{n} q_i$ are random, too. This implies that the multiplication is $\lfloor \frac{m-1}{2} \rfloor$-privately computable. This bound is strict. Furthermore, it can be shown that there is a perfect emulation.

5.3.4 Distributed RSA Key Generation

There are a handful of distributed RSA key generation schemes that – among other things – differ in their security assumptions, their computational complexity, the minimum number of participants required to generate a key pair and a signature respectively, and the way the distributed computation

is carried out. In this section, we give a brief description of each of these schemes. The various algorithms and some of their most important properties are listed in Table 5.1. Here m denotes the number of parties involved in key generation and k denotes the minimum number of parties required to generate a signature with a shared key. Computational costs are given in terms of the bit length ℓ of the RSA modulus. Our new algorithm, which is the subject of Section 5.4, fills a gap as the other schemes either have quadratic complexity or require the assistance of a third party which has to be trusted.

Author(s)	Parties	Signers	Costs†	RSA Cryptosystem
Boneh-Franklin [32]	$m \geq 3$	$k = \lceil \frac{n+1}{2} \rceil$	$O(\ell^2)$	standard
Cocks [57]	$m \geq 3$	$k = \lceil \frac{n+1}{2} \rceil$	$O(\ell^2)$	standard
Cocks [56]	$m = 2$	$k = 2$	$O(\ell^2)$	standard
Gilboa [102]				
Poupard-Stern [185]				
Boneh-Horwitz [185]	$m = 3$	$k = 2$	$O(\ell)$	multi-prime
Biehl-Takagi [28]	$m \geq 3$	$k = \lceil \frac{n+1}{2} \rceil$	$O(\ell^2)$	standard/multi-prime
Our Approach	$m = 2$	$k = 2$	$O(\ell)$	multi-prime

† ℓ = bit length of the generated RSA modulus.

Table 5.1: Algorithms for shared RSA key generation.

5.3.4.1 Boneh-Franklin Protocol

The Boneh-Franklin scheme is an m-party protocol, $m \geq 3$, which $\lfloor \frac{m-1}{2} \rfloor$-privately generates a standard RSA key pair. By means of this protocol we explain the consecutive stages that are required for each distributed RSA key generation algorithm. Constructing a (potential) RSA modulus out of the secret inputs of the parties is more or less canonical: the most simple way to do so is to write $N = (\sum_{i=1}^{m} p_i) \cdot (\sum_{i=1}^{m} q_i)$ where the p_i and q_i are secrets of the respective party. In this respect, the Boneh-Franklin protocol also served as a blueprint for subsequent protocols.

An outline of the protocol is shown in Figure 5.8. In the first two steps, a candidate modulus N is computed according to the formula given above.

Step 1: [Choosing random input] Each party picks ℓ bit random numbers p_i, q_i as secret inputs to the algorithm.

Step 2: [Computation of N] The parties jointly compute $N = pq$ where $p := \sum_{i=1}^{k} p_i, q := \sum_{i=1}^{k} q_i$ using (an extension) of the BGW protocol.

Step 3: [Trial divisions] The value of N is made public so that the parties can apply trial divisions up to a certain bound.

Step 4: [Primality test] In a distributed *Fermat test*, the parties check whether N is the product of exactly two primes. Essentially, this is done by testing

$$g^{(p-1)(q-1)} = g^{N+1-\sum_i (p_i+q_i)} \stackrel{?}{\equiv} 1 \mod N$$

for randomly chosen bases g.

Step 5: [Generation of d] The parties agree on a public exponent beforehand and compute the corresponding private exponent.

Figure 5.8: Outline of the Boneh-Franklin protocol.

The computation of N uses a slightly modified version of the BGW protocol which allows to construct N with only two rounds of communication. As this step is based on the BGW protocol, it is $\lfloor \frac{m-1}{2} \rfloor$-private.

The values p and q have a length of approximately ℓ bit. According to the Prime Number Theorem, the probability that an ℓ-bit number is prime is about $(\ln 2 \cdot \ell)^{-1}$. Since the numbers p and q are independently chosen at random, one has to expect $O(\ell^2)$ iterations until an N is found that is the product of exactly two primes. Otherwise N will not pass either the trial division or the distributed Fermat step.

Biehl and Takagi [28] show how to test the factors of N for primality separately (see below). The approach of Boneh and Horwitz [34] allows to speed up the computation for the case of three parties. A multi-prime RSA modulus of the form $(p_1 + p_2 + p_3)q_2r_3$ is used where the numbers q_2, r_3 are a priori prime. In the next section we show how to apply this idea to the case of two parties.

There is a very small fraction of numbers which passes the Fermat test of the Boneh-Franklin protocol without being a product of two primes. Therefore, further Fermat tests in the so-called *twisted group* $(\mathbb{Z}_n[x]/(x^2+1))^*/\mathbb{Z}_n^*$ were suggested [32], though neither of the present implementations makes use of them yet [160, 255, 13]. Both tests are $(m-1)$-private.

In the final step, the parties agree on a public exponent e, e.g. the standard value $e = 2^{16}+1$, and compute the private exponent without revealing the factorisation of N. This can be done in two ways. One of them is $(m-1)$-private, but leaks $\lceil \log_2(e) \rceil$ bits of information. The other method does not leak any information, but is only $\lfloor \frac{m-1}{2} \rfloor$-private (see [32] for details).

Practical Considerations Several independent implementations have already shown that the Boneh-Franklin scheme is indeed feasible in practice. Malkin *et al.* [160] report that the generation of a 1024-bit key among three servers takes about 1.5 or 6 minutes on a local area network or via the Internet respectively. Wright and Spalding [255] give a thorough analysis of how to tune parameters, for instance the trial division bound, for performance. The Boneh-Franklin protocol is also part of a distributed Java-based time stamping authority developed by NTT in cooperation with TU Darmstadt. Details are given in [13, 89, 243]. Experimental results show that the key generation phase for a 2048-bit key is less than half an hour when three or five servers on a local area network are involved.

Alternative Computation of N Cocks proposes an alternative method for the distributed computation of N given the p_i, q_i, both for the two-party and multi-party setting. We refer the reader to [56, 57] for details. An important downside of Cocks' protocols compared to the Boneh-Franklin method is the fact that their security only relies on a heuristic argument in contrast to the solid BGW foundation. Some attacks and countermeasures for Cocks' protocols have been proposed [29, 134], but the security properties remain unproven. A figure that is nevertheless interesting is a runtime estimate for the two-party case. Cocks [56] reports that the generation of a 512 bit key shared between two parties takes little more than one day (using Mathematica on a SPARC10 workstation).

Alternative Primality Test Biehl and Takagi [28] propose an alternative to step 4 based on quadratic fields that also extends naturally to the multi-

prime RSA case.[7] The security of the protocol relies on the discrete log problem in the class group of the quadratic order of discriminant $\Delta = -8N^2$ and a new intractability assumption "CDC" (computation of divisors of the conductor, see [28] for details).

An important property of the Biehl-Takagi protocol is the fact that each factor on the candidate modulus N is tested for primality separately. (Compare this to Boneh-Franklin's Fermat, test which only tells whether at least one of the factors is composite, but not which one.) At first sight, this seems to be only a subtle difference, but it allows a certain optimization as the parties may, instead of a two-factor candidate N, use a value which is the product of ℓ would-be primes (of bit length ℓ). The probability that at least two of these factors are in fact prime is approximately $1-2(1-(\ln 2\ell)^{-1})^{\ell-1}$. (For $\ell = 512$, this probability is about 50%.) On the one hand this reduces the number of expected iterations from $O(\ell^2)$ to $O(\ell)$. But, unfortunately, on the other hand the arguments grow in size to a bit length of ℓ^2 such that on the level of bit operations there is no efficiency gain.

5.3.4.2 Two-Party Protocols

Most parts of the Boneh-Franklin framework can be directly re-used in the two-party setting. The sole exception is the second step, i.e. the computation of N.[8] Similar to the original protocol, these algorithms have an expected running time quadratic in the bit length of the RSA modulus to be constructed as the factors are tested separately for primality. Several authors have proposed solutions for this problem. These schemes rely on the cryptographic primitive of oblivious transfer and a generalization of the quadratic residuosity assumption.

Oblivious Transfer An oblivious transfer (OT) scheme is an interactive two-party protocol which solves the following problem: A *merchant* who offers k secrets (bit-strings) wants to sell one of them to a buyer such that the privacy of the buyer with respect to the chosen index is kept and the remaining secrets are kept confidential as well. The names *All-or Nothing*

[7] The way their protocol implements step 2 slightly differs from Boneh-Franklin's proposal, but is still based on the BGW idea.

[8] It will turn out in Section 5.4.2.2 that step 4 has to be modified in the case of multi-prime RSA.

Disclosure of Secrets (ANDOS) protocol or *symmetrically private information retrieval* are used as synonyms for oblivious transfer schemes.

Definition 6 (Oblivious transfer/ANDOS protocol) *A protocol called* 1-out-of-k oblivious transfer *is a two-party scheme which allows a merchant to sell one out of k secrets $s_1, \ldots, s_k$ to a buyer satisfying the following properties. Let $i_0 \in \{1, \ldots, s\}$ denote the index chosen by the buyer.*

(1) The merchant obtains no information about i_0 during the protocol, i.e. the views

$$\text{View}_{\mathsf{merchant}}(s_1, \ldots, s_k, j),$$
$$\text{View}_{\mathsf{merchant}}(s_1, \ldots, s_k, i_0)$$

are indistinguishable for all $j \in \{1, \ldots, k\}$.

(2) The buyer obtains no information about the secrets other than s_{i_0}, i.e. the views

$$\text{View}_{\mathsf{buyer}}(s'_1, \ldots, s'_{i_0-1}, s_{i_0}, s'_{i_0+1}, \ldots, s'_k, i_0),$$
$$\text{View}_{\mathsf{buyer}}(s_1, \ldots, s_k, i_0)$$

are indistinguishable for all $s'_1, \ldots, s'_{i_0-1}, s'_{i_0+1}, \ldots, s'_k$.

Observe that indistinguishability in this definition may mean computational, statistical or perfect indistinguishability.

Quadratic and Prime Residuosity Problem The prime residuosity problem (PRP) is a natural generalization of the well-known quadratic residuosity problem (QRP) which can be stated as follows.

Definition 7 (Quadratic Residuosity Problem) *Given an odd composite integer N and an element $a \in \mathbb{Z}_N^*$ with Jacobi symbol $(a/N) = 1$, decide whether or not there exists $b \in \mathbb{Z}_N^*$ such that $a \equiv b^2 \mod N$ (i.e. a is a quadratic residue modulo N).*

For instance, the security of the Goldwasser-Micali probabilistic public-key encryption scheme is based on the assumed intractability of QRP. This assumptions says that every probabilistic polynomial-time algorithm to compute the predicate only succeeds with a probability that comes negligibly

(in the length of N) close to $1/2$, i.e. the advantage of obtaining the correct result compared with tossing a fair coin is negligible.

Now we can state the general problem for prime exponents p greater than 2. Both the scheme of Gilboa and the one we propose in the next section are based on the PRP intractability assumption.

Definition 8 (Prime Residuosity Problem) *Given an odd composite integer N, an odd prime p and an element $a \in \mathbb{Z}_N^*$, decide whether or not there exists $b \in \mathbb{Z}_N^*$ such that $a \equiv b^p \mod N$.*

Gilboa [102] has proposed three variants founded on different security assumptions. One of them is based on a 1-out-of-k oblivious transfer protocol. Another one computes N with the help of a homomorphic encryption scheme that is semantically secure under the *prime residuosity assumption.* The third one (the so-called oblivious polynomial evaluation, see [170]) uses a non-standard security assumption; we do not consider this variant any further.

Gilboa's OT Approach In the case of oblivious transfer,computations are done in the ring $\mathbb{Z}_r$ where r is a power of 2. Assume that Alice holds $a \in \mathbb{Z}_r$ and Bob holds $b \in \mathbb{Z}_r$. Then a multiplicative sharing ab can be converted into an additive sharing in the following way:

Alice and Bob go through r executions of an 1-out-of-2 OT protocol where Bob acts as the merchant and Alice acts as the buyer. Let $(a_{r-1}, \ldots, a_0)$ be the binary representation of a, i.e. $a = \sum_{i=0}^{r-1} 2^i a_i$. The following steps are executed for $i = 0, \ldots, r-1$: Bob chooses $s_i \in \mathbb{Z}_r$ uniformly at random and sets $t_i^0 := s_i, t_i^1 := 2^i b + s_i$. Alice buys one of these elements, namely $t_i^{a_i}$ depending on the value a_i of the appropriate bit in a's binary representation. It is easy to see that the numbers $x := \sum_{i=0}^{r-1} t_i^{a_i}$ and $y := \sum_{i=0}^{r-1} s_i$ (which can be computed by Alice and respectively Bob alone) satisfy the condition $x + y \equiv ab \mod r$.

In order to compute a candidate modulus $N = (p_1 + p_2)(q_1 + q_2)$, this protocol is applied twice to first obtain additive sharings of the mixed terms $p_1 q_2 = x^{(1)} + y^{(1)}$ and $p_2 q_1 = x^{(2)} + y^{(2)}$. Then Bob sends $y^{(1)} + y^{(2)} + p_2 q_2 \mod r$ to Alice who adds $x^{(1)} + x^{(2)} + p_1 q_1$ to obtain $N \mod r$ which equals N if r is chosen sufficiently large. The protocol is secure as the views of both parties can be simulated.

Gilboa's Homomorphic Encryption Approach This variant makes use of an asymmetric cryptosystem (ε, δ) with a the homomorphism property $\varepsilon(m_1)\varepsilon(m_2) = \varepsilon(m_1+m_2)$. This allows the parties to do certain computations on the encrypted values. These so-called homomorphic encryption schemes are explained in Section 5.4.1.1 in more detail as they are also an important building block for our scheme.

The expression $(p_1+p_2)(q_1+q_2)$ can be evaluated as follows: We assume that Alice sets up the cryptosystem (ε, δ) and provides Bob with the public key. Alice sends her encrypted inputs $\varepsilon(p_1), \varepsilon(q_1)$ to Bob who answers with

$$\varepsilon(p_1)^{q_2}\varepsilon(q_1)^{p_2}\varepsilon(p_2q_2).$$

Due to the homomorphic property, this value equals $\varepsilon(p_1q_2+q_1p_2+p_2q_2)$. After decrypting this value and adding p_1q_1 to it, Alice knows the value of $(p_1 + p_2)(q_1 + q_2)$ without having gained information (in an information-theoretic sense) about Bob's inputs p_2 and q_2. Alice's privacy is based on the security of the cryptosystem. Details are given in the context of the discussion in Section 5.4.1. In practice, this protocol is not applied for the inputs p_i, q_i themselves, but for their residue classes modulo a prime for a number of different moduli. The partial results are then recombined by Chinese remaindering.

Poupard-Stern Method The idea of Poupard and Stern [185] is similar to the previous method with respect to Chinese remaindering. However, the partial results modulo the prime moduli are computed differently. For each modulus t, the parties go through an ANDOS protocol with seller Alice and buyer Bob and another, similar protocol with the roles reversed. The ANDOS protocols allow to compute a generic function $f(d_1, d_2)$ on secret inputs d_i. In our concrete case, these inputs are pairs $(p_i, q_i) \in \mathbb{Z}_t \times Z_t$.

Let $(\alpha_i, \beta_i) \in \mathbb{Z}_t^* \times \mathbb{Z}_t, i = 1, 2$ be pairs of random numbers chosen by the respective party. At first, Alice sells $\{\gamma_d := \alpha_1 \cdot f(d_1, d) + \beta_1 : d\}$ where d ranges over the set of possible inputs for Bob. Bob buys the item $\alpha_1 \times f(d_1, d_2) + \beta_1$ whose index corresponds to his secret input d_2.

Alice and Bob then change their roles and Alice buys the item $\alpha_2 \cdot f(d_1, d_2) + \beta_2$. Finally, Alice and Bob simultaneously broadcast the values α_i, β_i and the item they have bought and ensure that they have locally computed the same value for $f(d_1, d_2)$.

It is easily seen that – given the security of the underlying OT scheme – this protocol guarantees privacy in the honest-but-curious scenario. Furthermore, it is shown in [185] that the protocol is also robust against a cheating party.

5.4 New Shared RSA Key Generation Protocol

This section describes a new and efficient method for shared multi-prime RSA key generation. Our setting is the honest-but-curious scenario as before. Throughout the whole protocol the parties have secure, i.e. private and mutually authenticated, communication channels available (e.g. SSL). Again we call the parties Alice (party 1) and Bob (party 2) and use indexed variables $v_i, i = 1, 2$, to denote secrets of the respective party unless stated otherwise.

5.4.1 Building Blocks

5.4.1.1 Homomorphic Encryption Scheme

Our method requires a public key cryptosystem with a probabilistic encryption function having a certain homomorphic property. Our notion of a homomorphic encryption scheme is a slight generalization of that given in [222].

Definition 9 (Homomorphic encryption scheme) *A* homomorphic encryption scheme *is a tuple* $(M, C, R, \varepsilon, \delta)$ *consisting of two groups* $(M, +)$ *and* $(C, \cdot)$*, a set* R*, and mappings* $\varepsilon : M \times R \to C$*,* $\delta : C \to M$ *satisfying the following:*

- $\forall m \in M \; \forall r \in R \;\; \delta(\varepsilon(m, r)) = m$,
- $\forall m_1, m_2 \in M \; \forall r_1, r_2 \in R \;\; \exists r_3 \in R$ *such that* $\varepsilon(m_1, r_1)\varepsilon(m_2, r_2) = \varepsilon(m_1 + m_2, r_3)$.

M and C are sets of messages and ciphertexts respectively, whereas R is used to randomize the encryption. Here we demand *semantic security* for ε.

Example 10 (Naccache-Stern cryptosystem) *Let* N *be a standard RSA modulus and* $t_i, i = 1, \ldots, n$*, small distinct prime factors of* $\varphi(N)$ *such that* $\gcd(t, \varphi(N)/t) = 1$ *where* $t := \prod_{i=1}^{n} t_i$*. Choose* $y \in \mathbb{Z}_N$ *such that*

$Y_i :\equiv y^{\varphi(N)/t_i} \not\equiv 1 \mod N$ for all i. A message $m \in M := \mathbb{Z}_t$ results in a ciphertext $\varepsilon(m,r) = r^t y^m \mod N$ where $r \in R := C := \mathbb{Z}_N$. To decrypt, the calculation of $c_i :\equiv \varepsilon(m,r)^{\varphi(N)/t_i} \equiv Y_i^m \mod N$ for each i is followed by a table lookup of $m \mod t_i$ in the set $\{(Y_i^s, s) : s \in \mathbb{Z}_{t_i}\}$.

The security of the Naccache-Stern cryptosystem is proved in [169] under the *higher residuosity assumption*. It is a generalization of the Benaloh [25] cryptosystem ($n = 1$) which is semantically secure under the *prime residuosity assumption*. The Benaloh cryptosystem is in its turn a generalization of the Goldwasser-Micali [109] scheme ($t = 2$ and Jacobi symbol $(y/N) = 1$) which is semantically secure under the *quadratic residuosity assumption*. Benaloh cryptosystems, which we use extensively in our protocol, can be set up efficiently [25].

5.4.1.2 Sharing Conversion

Homomorphic encryption schemes have proven to be quite helpful for shared computations with secret inputs. We have already seen an example from [102] in Section 5.3.4.2. We will now introduce the notion of additive and multiplicative secret sharings based on homorphic encryption.[9]

Definition 11 (Notion of secret sharing) *Let M be a ring and $s \in M$ a secret value. Assume that party i $(i = 1, 2)$ holds $a_i, m_i \in M$ such that*

$$a_1 + a_2 = s = m_1 m_2.$$

The pairs (a_1, a_2) and (m_1, m_2) are called an additive *resp. a* multiplicative sharing *of s.*

Our protocol needs a subroutine to convert an additive sharing into a multiplicative one: The algorithm `add2mult` described in Figure 5.9 is inspired by the trick mentioned above. We will apply `add2mult` to secrets in the ring $M := \mathbb{Z}_t$. Therefore, we sometime write $\texttt{add2mult}_t$ to make the modulus explicit. For $m \in M$ and $c \in C$, we write c^m as a shorthand for c^u where $u \in \mathbb{Z}$ is the smallest non-negative representative of the residue class m.

Without loss of generality, we assume that it is always Alice who sets up the homomorphic encryption scheme $(M, C, R, \varepsilon, \delta)$ needed in the protocol. By (a_1, a_2) we denote an existing additive sharing of s. Messages exchanged

[9] From now on, we omit the second argument of ε for the sake of simplicity.

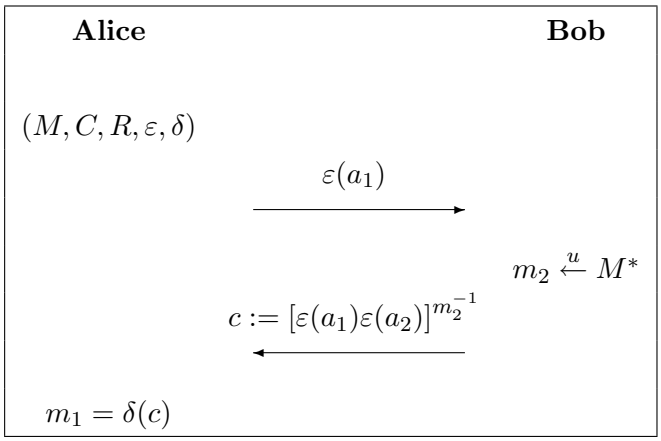

Figure 5.9: `add2mult` sharing conversion protocol.

in the protocol are shown in Figure 5.9. The notation $m_2 \xleftarrow{u} M^*$ means that m_2 is chosen from M^* uniformly at random (and independently from s). At the end of the protocol, the parties have obtained the multiplicative sharing (m_1, m_2). The following proposition states that Alice cannot learn anything about s if s is a unit. However she will find out whether indeed $s \in M^*$. Fortunately, this does no harm because $s \notin M^*$ is rejected in our protocol anyway (see below).

Proposition 12 (Bob's Privacy for `add2mult`) *Let M be a ring and $s \in M^*$ an additively shared secret value. Then Alice gains no information about s during the protocol `add2mult`.*

Proof. To model the protocol we introduce M^*-valued random variables X, Y_1, Y_2. Let prob_X denote the probability distribution of s and prob_{Y_i} denote the probability distribution of m_i. Then

$$\begin{aligned}\text{prob}(X = s \wedge Y_1 = m_1) &= \text{prob}(X = s \wedge Y_2 = sm_1^{-1}) \\ &= \text{prob}_X(s) \cdot \text{prob}_{Y_2}(sm_1^{-1}) = \text{prob}_X(s)/|M^*|\end{aligned}$$

since X and Y_2 are independent and Y_2 is uniformly distributed by assumption. Using the same argument, we get

$$\text{prob}_{Y_1}(m_1) = \sum_{s \in M^*} \text{prob}(X = s \wedge Y_2 = sm_1^{-1}) = \frac{1}{|M^*|}.$$

Substituting the second into the first equation we deduce that X and Y_1 are independent which means that their mutual information is always zero. □

Observe that Bob's privacy is guaranteed in an information theoretic sense while Alice's relies on the security of the homomorphic encryption scheme. As a compensation, they could change roles in practice each time `add2mult` is used. This may also be advisable with regard to load-balancing.

Proposition 13 (Alice's Privacy for `add2mult`) *Let M be a ring and $s \in M^*$ an additively shared secret value. The transcript of the view of an `add2mult` execution can be simulated for Bob, therefore Alice's privacy is preserved if the homomorphic encryption system used in the protocol is semantically secure.*

Bob's view during the protocol with output $\text{Output}^{\texttt{add2mult}}(a_1, a_2) = (m_1, m_2)$ is

$$\text{View}_{\text{Bob}}^{\texttt{add2mult}}(a_1, a_2) = (a_2, m_2, \varepsilon(a_1)).$$

Given Bob's secret input a_2 and his output m_2, the simulator picks $x \in M$ uniformly at random. The sequence $(a_2, m_2, \varepsilon(x))$ cannot be distinguished from a true protocol transcript as this would contradict the semantic security of the cryptosystem (ε, δ). □

We briefly show how to define a protocol `mult2add` to convert in the reverse direction.[10] Assume that again Alice sets up $(M, C, R, \varepsilon, \delta)$. The messages exchanged are $\varepsilon(m_1)$ and $c := \varepsilon(m_1)^{m_2}\varepsilon(-a_2)$ where Bob chooses $a_2 \overset{u}{\leftarrow} M$. Alice obtains her share a_1 by decrypting c. The following propositions, which can be proved analogously to Proposition 12 and 13, will be used in Section 5.4.2.2:

Proposition 14 (Bob's Privacy for `mult2add`) *Let M be a ring and let $(m_1, m_2) \in M^* \times M$ be a multiplicative sharing. Then Alice gains no information about $m_1 m_2$ during the protocol `mult2add`.*

Proposition 15 (Alice's Privacy for `mult2add`) *Let M be a ring and $s \in M^*$ an additively shared secret value. The transcript of the view of an `mult2add` execution can be simulated for Bob, therefore Alice's privacy is preserved if the homomorphic encryption system used in the protocol is semantically secure.*

[10] Boneh-Franklin and Gilboa use BGW and oblivious transfers respectively for the same purpose.

5.4.1.3 Distributed Sieving

Distributed sieving is an elegant way to combine the random choice of input data with trial divisions (steps 1 and 2 of the Boneh-Franklin framework in Section 5.3.4.1). Similar techniques have been used in [32, 102] to reduce the number of iterations since the likelihood that a candidate modulus is the product of exactly two primes is greater if one excludes small divisors.

Figure 5.10 shows a description of `distSieve` in pseudo-code from the viewpoint of party i. This subroutine allows to negotiate a secret $s \in \mathbb{Z}$ represented by an additive and a multiplicative sharing such that s is not divisible by elements of a set $P \in \mathbb{P}$ where $\mathbb{P}$ is the set of prime numbers. Usually, P is the set of all primes up to a certain bound.

```
distSieve(P)
1  for t ∈ P
2        m_i^(t) ←u Z_t^*
3        a_i^(t) ←mult2add(m_i^(t))
4  a_i ←CRT((a_i^(t))_{t∈P})
5  m_i ←CRT((m_i^(t))_{t∈P})
6  return (a_i, m_i)
```

Figure 5.10: Subroutine `distSieve`.

For each $t \in P$, Alice and Bob choose random inputs $m_i^{(t)} \in \mathbb{Z}_t^*$ and convert the multiplicative sharing $(m_1^{(t)}, m_2^{(t)})$ in an additive sharing $(a_1^{(t)}, a_2^{(t)})$ using `mult2add`. As the inputs of both parties are units modulo t, this guarantees that the resulting m_i are units, too. At the end of `distSieve`, the parties combine their shares $a_i^{(t)}, t \in P$ and $m_i^{(t)}, t \in P$ by means of the Chinese Remainder Theorem to a_i and m_i respectively.

We use the notation $\prod P$ as a shorthand for $\prod_{t \in P} t$. Then the subroutine `distSieve` returns two non-negative integers less than $\prod P$. It *implicitly* returns an integer $0 < s < 2 \prod P$ such that $s \equiv a_1 + a_2 \equiv m_1 + m_2 \mod \prod P$ and s is relatively prime to $\prod P$. Here we use the term "implicitly" to

express that the s is a secret value shared by the parties, but never computed explicitly.

Proposition 16 (Privacy of `distSieve`) *During the protocol `distSieve`, the privacy of both parties concerning $a_1 + a_2 \equiv m_1 m_2 \mod \prod P$ is preserved provided the prime residuosity assumption is true.*

Proof. The protocol consists of $|P|$ interactions where a random number $m_i^{(t)}$ is picked and the secret $m_1^{(t)} m_2^{(t)}$ is converted to an additive sharing using `mult2add`. According to Propositions 14 and 15, the sub-protocol `mult2add` preserves the privacy of both parties. Apart from the invocation of `mult2add`, all steps of the protocol are local computations of only one party. □

5.4.2 The New Protocol

The key idea of the new scheme is to use a multi-prime RSA modulus instead of the usual two-factor RSA modulus. This may seem counter-intuitive at first sight as each additional factor will increase the expected number of iterations even more: We have seen that the complexity of distributed key generation for a standard modulus of length ℓ is $O(\ell^2)$ since the two factors are tested simultaneously for their primality. Using a modulus of, for instance, three factors would obviously raise the complexity to $O(\ell^3)$.

However, this statement only holds for moduli of the form $(p_1 + p_2)(q_1 + q_2)(r_1 + r_2)$ where the p_i, q_i, and r_i are secret values selected at random by the respective party. Things can be made simpler by composing N according to

$$N := pq_1q_2 := (p_1 + p_2)q_1q_2$$

where the p_i are arbitrary random numbers, but the q_i are *primes*. Now that two of the factors are *a priori* prime, one can in fact find a product of three primes in expected time $O(\ell)$.

A similar idea for the three party scenario is described in [34] where the modulus is composed according to $(p_1 + p_2 + p_3)q_1q_2$. However, in this case, party 1 and 2 have a certain advantage over party 3 as each of them knows a proper factor of the modulus.

In the following two sections, we describe how to compute a candidate modulus and how to organize a secure primality test in a distributed way.

```
candidate(P, P')
1  (p_i, m_i) ← distSieve(P)
2  for t ∈ P' \ P
3        m_i^(t) ←add2mult(p_i mod t)
4        if m_1^(t) = 0 then goto 1
5  m'_i ←CRT(m_i, (m_i^(t))_{t∈P'\P})
6  q_i ← randomPrime(2^μ)
7  publish m'_i q_i mod ∏P'
8  return m'_1 q_1 m'_2 q_2 mod ∏P'
```

Figure 5.11: Computing a candidate modulus.

5.4.2.1 Computing a Candidate Modulus

Our algorithm to generate a candidate multi-prime RSA modulus is shown in Figure 5.11. It has two arguments $P \subset P' \subset \mathbb{P}$ subject to certain technical conditions (see below).

At first, the parties use `distSieve` to implicitly generate a random invertible residue class modulo $\prod P$. Let p denote the smallest non-negative representative of this residue class. The multiplicative sharing (m_1, m_2) of p is then extended to a sharing (m'_1, m'_2) of $p \bmod \prod P'$ in lines 2–5. This is a technique similar to that in `distSieve`, however the inputs $p_i \bmod t$ are fixed now (and might not be invertible modulo t). As a consequence, we have to cope with the possibility that $t \mid p$ for a certain $t \in P' \setminus P$. But this can be always detected as the party which takes on Alice's role in the `add2mult` protocol outlined in Figure 5.9 would obtain the result 0 and thus could declare "failure". In this case, the algorithm is completely restarted (see [102] for possible optimizations).

Having successfully passed all tests of line 4, each party separately computes an integer m'_i with the Chinese Remainder Theorem (CRT) and generates a prime number, which will become the second and third factor respectively of the RSA modulus in case $m'_1 m'_2$ is in fact prime. The CRT

computations yield an m_i' such that

$$m_i' \equiv m_i \mod \prod P$$
$$m_i' \equiv m_i^{(t)} \mod t \quad \forall t \in P' \setminus P.$$

Furthermore, in lines 6 and 7 of `candidate` Alice and Bob generate primes q_i of bit length μ and publish the values $m_i' q_i \bmod P'$ to the other party. These values are a multiplicative sharing of a candidate modulus modulo $\prod P'$. Here, the purpose of q_i is to *blind* the secret m_i' from the other party in order to conceal the value of $p \mod P'$.

The following theorem lists the properties of the outcome of `candidate` and makes the choice of P and P' explicit. Theorem 18 guarantees that the privacy of both parties is preserved during the protocol.

Theorem 17 (Correctness of `candidate`) *Let ℓ and μ be two integer constants and let $P \subset P' \subset \mathbb{P}$ be two sets subject to the following conditions:*

- *$P = \{2, 3, 5 \ldots\}$ is the minimal set of all consecutive primes such that*

$$\prod P > 2^{\ell-1}.$$

- *P' is chosen such that*

$$\prod (P' \setminus P) > 2^{2\mu+1}$$

.

Then – using `candidate` – the parties find a positive integer pq_1q_2 of bit length at most $2\mu + \ell + 1$ where the q_i are prime and p is not divisible by any number in P'.

Proof. After the distributed sieving step, the parties have agreed on a value $p \equiv p_1 + p_2 \equiv m_1 m_2 \mod \prod P$ where $0 \leq p_1, p_2, m_1, m_2 < \prod P$. The parties know the additive sharing $(p_1 \bmod t, p_2 \bmod t)$ of $p \bmod t$ for each $t \in P' \setminus P$. These additive sharings are converted to multiplicative sharings $(m_1^{(t)}, m_2^{(t)})$ which satisfy $m_1^{(t)} m_2^{(t)} \equiv p \mod t$ for all $t \in P' \setminus P$. Consequently, we have $m_1' m_2' \equiv p \mod P'$.

Finally, let N denote the smallest non-negative representative of the residue class $m_1' q_1 m_2' q_2$ modulo P'. Then obviously $N \equiv pq_1q_2 \mod \prod P'$ as $m_1' m_2' \equiv p \mod P'$, but due to the size constraints on the factors

$$0 \leq pq_1q_2 = (p_1 + p_2)q_1q_2 < 2 \prod P \cdot 2^{2\mu} < \prod P',$$

we even have equality $N = pq_1q_2$. The bit length of the modulus$N$ is at most $2\mu + \ell + 1$. □

Theorem 18 (Privacy of `candidate`) *During the execution of the protocol `candidate`, Alice (Bob) learns nothing about $q_2 \bmod t$ ($q_1 \bmod t$) for an arbitrary $t \in P'$.*

Proof. Alice surely knows $m_2^{(t)} q_2 \bmod t$. Although $q_2 \bmod t$ is not uniformly distributed in $\mathbb{Z}_t^*$, the product in fact is since $m_2^{(t)}$ was chosen accordingly. A similar argument shows that Bob's view during the protocol can be simulated. Therefore, Alice's privacy is also assured. □

5.4.2.2 Primality Test

The idea of the primality test for a three-factor modulus is similar to the one described in Section 5.3.4.1. We set $\psi_3(N) := (p_1+p_2-1)(q_1-1)(q_2-1)$ and use the criterion $g^{\psi_3(N)} \equiv 1 \mod N$ to check whether N is of the desired form. In case p is prime, $\psi_3(N)$ equals $\varphi(N)$ and the condition is fulfilled.

We have seen that – in the standard RSA case – the exponent $\psi_2(N) = (p_1 + p_2 - 1)(q_1 + q_2 - 1)$ is shared additively between the parties according to $\psi_2(N) = (N + 1 - p_1 - q_1) - (p_2 + q_2)$. Things are not so easy any more for a three-factor modulus since expanding the expression $\psi_3(N)$ yields two mixed terms, namely the multiplicatively shared secrets q_1p_2 and $(p_1+q_1)q_2$. Fortunately these can be converted to additive sharings (a_1, a_2) and (b_1, b_2) respectively as we will show below.

The Fermat Criterion Loosely spoken, during the Fermat test random bases $g \in \mathbb{Z}_N^*$ are picked and the following condition is checked. The left resp. right hand side of the expression can be computed solely by the first resp. second party.

$$g^{N-p_1q_1+p_1+q_1-1-a_1-b_1} \stackrel{?}{\equiv} g^{p_2q_2-p_2-q_2+a_2+b_2} \mod N.$$

This test amounts to checking whether $g^{\psi_3(N)} \equiv 1 \mod N$.

If p is prime, the Fermat test obviously succeeds for all $g \in \mathbb{Z}_N^*$ since $\psi(N)$ equals $\varphi(N)$. The probability that an element chosen at random from $\mathbb{Z}_N$ is a unit is overwhelming (on the order of $1 - 4^{-\ell\mu}$), so we do not have to treat the case $g \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$ separately as it is very unlikely that a correct N is rejected due to the choice of g.

Theorem 19 (Security of Fermat Test) *An easy simulation argument shows that Fermat tests do not leak information apart from their result if N is the product of exactly three different primes.*

Preparing the Input Data We briefly explain how to treat the multiplicatively shared inputs. Consider for instance (q_1, p_2). Let P'' be a set of odd primes such that

$$\prod P'' > (\prod P + 2^{\mu})2^{\mu}.$$

Combining the results of `mult2add` over $\mathbb{Z}_t, t \in P''$, the parties obtain an additive sharing (a_1, a_2) over $\mathbb{Z}_{\prod P''}$ without leaking information. The subroutine that calls `mult2add` for all $t \in P''$ and finally combines the results by CRT is abbreviated by $\texttt{mult2add}_{\prod P''}$ in Figure 5.12. Due to the size of the shares, one has $q_1 p_2 = a_1 + a_2 - \gamma \prod P''$. Here, $\gamma \in \{0, 1\}$ can be easily calculated by revealing the values $a_i \bmod 2$ (we can assume without loss of generalisation that p_2 is always even). Similarly let $(p_1 + q_1)q_2 = b_1 + b_2 - \eta \prod P''$. The parties first compute their respective exponents and then test bases chosen at random. Depending on the desired confidence level, the number Γ of bases, which must satisfy the condition, can be chosen flexibly. Theorem 20 guarantees that the probability to err is at most $4^{-\Gamma}$. The parties select bases at random until they have found and checked Γ bases g for which $(g/N) = (g/q_1) = (g/q_2) = 1$ holds. This is the main loop in Figure 5.12

Success Probability The following theorem states that the Fermat test will fail with a probability at least $3/4$ if N is not of the desired form. Therefore the probability that a p which is not prime sustains Γ successive Fermat tests is at most $4^{-\Gamma}$. In practice, the value of Γ should be chosen according to the confidence parameter used for the probabilistic test for the generation of q_i. A typical confidence level is $1 - 2^{-80}$, which is used for instance by the FlexiProvider as a default for prime number generation, i.e. $\Gamma = 40$.

For technical reasons which are due to our proof, the set of candidate bases is restricted to the subgroup

$$G := \{x \in \mathbb{Z}_N^* : (x/p) = (x/q_1) = (x/q_2) = 1\}$$

```
primalityTestP(p_i, q_i, N)
 1  if gcd(q_i − 1, N) ≠ 1 return ‘‘reject’’
 2  a_i ←mult2add_{∏ P''}(q_1, p_2)
 3  b_i ←mult2add_{∏ P''}(p_1 + q_1, q_2)
 4  x_i ← { N − p_1 q_1 + p_1 + q_1 − 1 − a_1 − b_1 + γ ∏ P''   :  i = 1
           { p_2 q_2 − p_2 − q_2 + a_2 + b_2 − η ∏ P''          :  i = 2
 5  for j = 1 ... Γ
 6      g ← randomNumber(N)
 7      if (g/N) ≠ 1 goto 6
 8      publish (g/q_i)
 9      if (g/q_1) ≠ 1 or (g/q_2) ≠ 1 goto 6
10      publish y_i := g^⌊x_i/8⌋ mod N
11      if y_1 ≠ y_2 return ‘‘reject’’
12  return ‘‘p probably prime’’
```

Figure 5.12: Primality test for factor p.

of $\mathbb{Z}_N^*$. The elements of G are squares modulo q_1q_2, but not necessarily modulo p (only in case p is prime). Membership in G can be checked efficiently: Given $g \in \mathbb{Z}_N$, each party checks that $(g/q_i) = 1$ and that $(g/N) = 1$, too.

As an additional requirement, we restrict the set of generated moduli to numbers where the factors p, q_1, and q_2 are congruent 3 modulo 4. The q_i are chosen appropriately by the parties. The constraint $p \equiv 3 \mod 4$ can be enforced in two ways. The first one is simple, yet inefficient as it discards all moduli N where $N \equiv 1 \mod 4$. As an alternative the parties may exclude 2 from the set P and fix their values modulo 4. This requires only a slight modification of `distSieve`. We use the following notation: For a positive integer n and a finite Abelian group A, $\mathrm{QR}(n)$ denotes the subgroup of quadratic residues in $\mathbb{Z}_n^*$, $\mathrm{J}^+(n)$ denotes the subgroup of elements having

Jacobi symbol 1, and $\mathrm{R}_n(A)$ denotes the subgroup $\{h \in A : h^n = 1\}$ of n-th roots of unity. Consequently, G is isomorphic to $\mathrm{QR}(q_1q_2) \times \mathrm{J}^+(p)$. We write $p = 2p' + 1, q_i = 2q_i' + 1$ (the p' and the q_i' are odd). From now on, we abbreviate $\psi_3(N)$ by $\psi(N)$.

Theorem 20 (Subgroup Index) *Let $N = pq_1q_2$ where the q_i are prime and all of the three factors are congruent 3 modulo 4. Let $G := \{x \in \mathbb{Z}_N^* : (x/p) = (x/q_1) = (x/q_2) = 1\}$ and $H = \mathrm{R}_e(G)$ where $e := \psi(N)/8$.*

If p is prime, then $H = G$, otherwise H is a proper subgroup of G and the index $[G : H]$ is at least 4.

The Fermat test in the group G works as follows: In each iteration, $g \in G$ is selected uniformly at random. If $g^e \bmod N$ equals 1, "success" is declared. Otherwise the test stops indicating "failure". Since the index of H in G is at least 4, this happens with probability at least 3/4.

Proof. We first treat the case where p is prime (1.) and then consider a composite p which is and is not squarefree (2b. and 2a.).

1. Assume that p is prime. Then $g^e \mod p \equiv (g^{(p-1)/2})^{q_1'q_2'} \equiv (g/p)^{q_1'q_2'} \equiv (g/p) \equiv 1 \mod p$ using the property of the Legendre symbol and the fact that $q_1'q_2'$ is odd. Similarly, $g^e \equiv 1 \mod q_i$ for $i = 1, 2$. Therefore, $g \in G$ implies $g \in H$.

Now assume that p is composite, say $p = \prod_{i=1}^k p_i^{d_i}$ for distinct primes p_i and $k \geq 2$ or $\sum_{i=1}^k d_i \geq 2$. We will show that $G \setminus H$ is non-empty.

2a. We first treat the case where one exponent d_i is greater than 1. Without loss of generality, we may assume that $d_1 \geq 2$. This implies that p_1 divides $\varphi(N) = (q_1 - 1)(q_2 - 1)\prod_{i=1}^k p_i^{d_i-1}(p_i - 1)$. However, p_1 does not divide q_i' $(i = 1, 2)$ as the parties have checked beforehand that $\gcd(N, q_i') = 1$ which in turn implies $\gcd(q_i', p_1) = 1$ $(i = 1, 2)$.

The subgroup of Z_N^* which is isomorphic to $Z_{p_1^{d_1}}^*$ is cyclic of order $p_1^{d_1-1}(p_1 - 1)$. In particular, Z_N^* contains an element g of order p_1 which is congruent to 1 modulo $q_1q_2\prod_{i=2}^k p_i^{d_i}$. Since $(g/p) = (g/p_1^{d_1}) = (g/p_1)^{d_1} = (g/p_1)^{p_1d_1} = (g^{p_1}/p_1)^{d_1} = (1/p_1)^{d_1} = 1$, we have $g \in G$.

Assume that $g^e \equiv 1 \mod N$. This implies that p_1 divides $2e = 2p'q_1'q_2'$. However, p_1 does neither divide $2p' = p_1^{d_1}\prod_{i=2}^k p_i^{d_i} - 1$, nor

one of the q_i' (since $\gcd(N, q_i') = 1$). This yields a contradiction and actually proves $g \notin H$.

One can show that $[G : H] \geq p_1^{d_1 - 1}$ as follows. Let x be a generator of $Z^*_{p_1^{d_1}}$ and set $y \equiv x^2 \mod p_1^{d_1}$ and $y \equiv 1 \mod q_1 q_2 \prod_{i=2}^{k} p_i^{d_i}$. Since y is a square, it is an element of G. We consider the subgroup $\langle y \rangle$ generated by y. We have shown that H does not contain elements of order p_1, so $\langle y \rangle \cap H$ is trivial. From elementary group theory, we have the isomorphism $(\langle y \rangle H)/H \cong \langle y \rangle / (\langle y \rangle \cap H) \cong \langle y \rangle$. The inclusions $H < \langle y \rangle H \leq G$ imply $[G : H] \geq \mathrm{ord}(y) = p_1 > 4$.

A number p that has been constructed according to the algorithm, is not divisible by small primes, say up to a bound of 10^5 for example. As a consequence, candidates which are divisible by a square are sorted out with a fairly high probability ($\geq 1 - 10^5$).

2b. We are left with the case where p is squarefree and $k \geq 2$. We use the Chinese remainder theorem to find an element g which is congruent to 1 modulo $q_1 q_2$ and modulo all p_i for $i \geq 3$. Furthermore, let $g \equiv -1 \mod p_1$, and $g \equiv a \mod p_2$ where $a \in \mathbb{Z}^*_{p_2}$ is an arbitrary element such that $(a/p_2) = (-1/p_1)$. This guarantees that $(g/p) = \prod_{i=1}^{k} (g/p_i) = 1$. Obviously, $g \in G$. But since e is odd and $g^e \equiv -1 \mod p_2$ by construction, $g^e \not\equiv 1 \mod N$ and thus $g \notin H$.

We note that in this case, the index of H in G is at least $[G : \mathrm{QR}(N)]$ as $H \leq \mathrm{QR}(N) < G$. ($H \leq \mathrm{QR}(N)$ because $1 = (1/s) = (g^e/s) = (g/s)^e = (g/s)$ for all divisors s of N.) The index $[G : \mathrm{QR}(N)]$ equals $[\mathrm{J}^+(p) : \mathrm{QR}(p)]$ which is 2^{k-1}.

This estimate can be improved for $k = 2$. As $p = p_1 p_2 \equiv 3 \mod 4$, one of the prime factors is congruent 1 modulo 4, say p_1. We have

$$H = \mathrm{R}_e(G) \cong \mathrm{R}_e(\mathrm{J}^+(p)) \times \mathrm{R}_e(\mathrm{QR}(q_1 q_2)) = \mathrm{R}_e(\mathrm{J}^+(p)) \times \mathrm{QR}(q_1 q_2)$$

since $g^e \equiv 1 \mod N$ for all $g \in \mathrm{QR}(q_1 q_2)$. Consider the Hasse diagram of $\mathrm{J}^+(p)$ which is depicted in Figure 5.13. Observe that $\mathrm{R}_e(\mathrm{J}^+(p)) = \mathrm{R}_e(\mathrm{QR}(p))$ since $g^e \equiv 1 \mod p_i$ implies $(g/p_i) = 1$ (see the argument above).

We show that $\mathrm{R}_e(\mathrm{J}^+(p)) \lneq \mathrm{QR}(p_1) \times \mathrm{R}_e(\mathrm{QR}(p_2))$: Let x be a generator of $\mathbb{Z}^*_{p_1}$. Chose $g \in \mathbb{Z}^*_p$ such that $g \equiv x^2 \mod p_1$ and $g \equiv 1 \mod p_2$.

Then $g \in \mathrm{QR}(p_1) \times \mathrm{R}_e(\mathrm{QR}(p_2))$, but $g \notin \mathrm{R}_e(\mathrm{J}^+(p))$ since the order of g, which is the even number $(p_1-1)/2$, does not divide e, which is odd. Therefore, $[G : H] = [\mathrm{J}^+(p) : \mathrm{R}_e(\mathrm{J}^+(p))] \geq [J^+(p) : \mathrm{QR}(p)][\mathrm{QR}(p) : \mathrm{QR}(p_1) \times \mathrm{R}_e(\mathrm{QR}(p_2))] \geq 4$ for $k = 2$, too. □

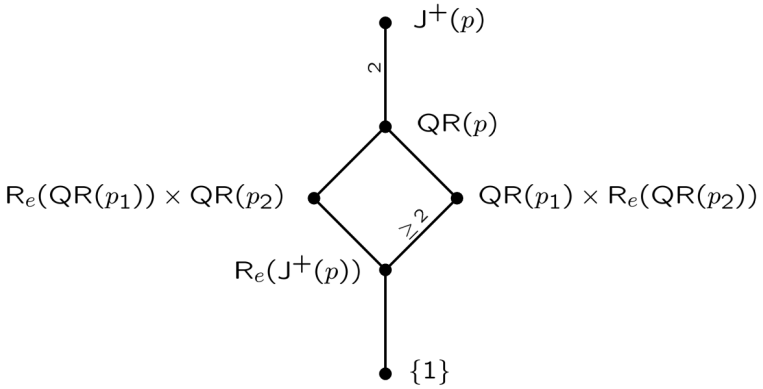

Figure 5.13: Hasse diagram of $\mathrm{J}^+(p)$ (proof case 2b. with $k = 2$).

Estimating the Number of Iterations Let us briefly estimate the expected number of iterations until a suitable 3-prime modulus is found. Our algorithm proceeds in three stages. The first stage is `distSieve` after which the parties have ensured that p is not divisible by a prime in P. During the second stage, the sharings mod $\prod P$ are "lifted" to sharings mod $\prod P'$ simultaneously checking whether p is divisible by a factor in $P' \setminus P$. If this is true the parties have to start over again. Otherwise they locally subject p to trial divisions up to a certain bound (in our experiments we chose 50,000). If no "small" divisors are found, the parties enter stage three, i.e. the Fermat `primalityTestP` test.

One can use a result due to DeBruijn (see [32]) to estimate the probability of p being prime under the condition that it has no divisors less than B. This probability is approximately $2.57 \cdot \ln(B)/\ell$ where ℓ is again the bit length of p. Having withstand the trial divisions, this value is about $1/14, 1/19$, and $1/28$ for $\ell = 384, 512$, and 768 respectively. The reciprocal is an estimate for the number of executions of `candidate` plus subsequent local trial divisions.

For further illustration, we provide some sample values for the variables P, P', P'' when chosen according to the above formulae. We restrict ourselves to the case $\ell = \mu$. Table 5.2 shows some typical parameters for ℓ between

384 and 768. Note that the actual length of the modulus may be a few bits longer than 3ℓ. With our prototype the average time measured for one run of `candidate` including local trial divisions is about 25 seconds for $\ell = \mu = 384$ and 35 seconds for $\ell = \mu = 512$. The time for checking one base in the Fermat test is about 2 seconds. In an experiment we created 1,500 moduli of bit length 1548. Of those 36% failed the trial division step and the rest entered the Fermat test. We found that approximately 3% of the moduli consisted of three primes ($\Gamma = 10$). Choosing a higher value does not significantly change computation time. In order to find a suitable base for the Fermat test on average one has to reject one of two candidates as $(g/p_i) = -1$ may occur with probability 1/2. A rough estimate shows that generating a suitable 1548 bit modulus takes less than half an hour. And note, that our implementation is a prototype which can be optimized in many respects, e.g. by using pre-computed values or applying a stronger form of parallelism (at the moment, the processes loose time for synchronisation as only the `mult2add` and `add2mult` operations for one candidate are carried out in parallel.

$\ell = \mu$	$\lvert P\rvert$	$\max P$	$\lceil \lg_2(\prod P)\rceil$	$\lvert P'\rvert$	$\max P'$	$\lvert P''\rvert$	$\max P''$
384	60	281	385	145	839	104	571
512	76	383	519	184	1103	132	751
768	105	571	777	258	1637	183	1103

Table 5.2: Typical parameters for `candidate`.

5.4.3 Summary

We have just presented our new RSA key generation scheme for the two-party setting and shown its security in the honest-but-curious model. A general discussion about the security of multi-prime RSA follows below. As a security assumption we postulated that the homomorphic cryptosystem we used should be semantically secure. The necessary instances of the Benaloh cryptosystem (or another) can be easily set up in advance and can be re-used in different passes of the protocol. Since breaking a single cryptosystem would only leak a value in $\mathbb{Z}_t$, we consider 1024 bit to be sufficient for the Benaloh moduli. Strictly speaking, we have omitted the last step in the private generation, namely the computation of a shared private exponent.

Known techniques can be used for this task. They are efficient and preserve privacy also in the two-party setting (under the same assumptions as they only require sharing conversions). We refer the reader to [32, 102].

Various attacks on (standard) RSA have been looked for and studied for quite a long time (see e.g. [30] for a survey). However, no methods are known which exploit the special structure of a 3-prime modulus. Some known attacks can be generalized to multi-prime RSA, but become even less effective [125]. As of this writing, 3-prime moduli of at least 1024 bit are considered as secure as 2-prime moduli of the same length [152]. Even if one does not agree and one wants to make a very defensive and careful choice, one may set $\ell := \mu := n/2$ in situations where one would accept a 2-prime RSA modulus of length n bit.

A general advantage of multi-prime moduli N is the fact that private key computations modulo N can be further accelerated using the CRT while parties that only know the public key do not notice the special structure of the modulus. Compared to the Boneh-Franklin setting where a single party cannot do computations modulo a factor of N, in our case an (asymptotic) speedup of 3 is possible as each party knows factors of length $n/3$ bit and $2n/3$ bit (see [35] for details).

5.5 Further Applications

In this section we show two more applications of threshold cryptography. We consider an email scenario where RSA (or ElGamal) is used as an encryption scheme. The basic ideas and the mathematics of RSA threshold signatures and threshold decryption are quite similar. The first application is a vacation replacement scenario, the second one devotes itself to the protection of enterprise gateways for secure email. For both applications prototypes are available as a proof of concept.

5.5.1 Delegating Decryption Keys for Secure Email

This section addresses the problem outlined in Section 3.2.5, namely the question of how to cope with encrypted email which should be temporarily forwarded to another person (or another mailbox of the same person). Currently, there are only two alternatives which are both disadvantageous for the private key holder: Storing the key pair on an online system allows

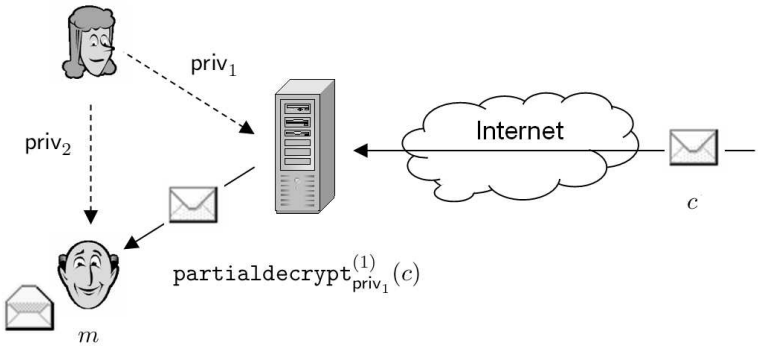

Figure 5.14: Threshold cryptography solution for vacation replacement problem.

seamless decryption and the revocation of capabilites when necessary, but makes the key vulnerable to online attacks (e.g. password-guessing is possible for webmail systems). On the other hand, releasing the private key from the sole control of the owner and the safe place it is stored is risky as such a step cannot be easily revoked. Plus, it is virtually impossible to track what the key is used for (think of keys that are also used for authentication or digital signatures).

Figure 5.14 shows our solution to this problem. Here Alice is the one who wants to delegate the ability of accessing her encrypted email temporarily to Bob. Let $\langle \mathsf{priv}, \mathsf{pub} \rangle$ denote her key pair. During setup, Alice uses the `Split` class of our software package to generate two shares of her private key priv. These shares are denoted by priv_1 and priv_2 in the drawing. Alice stores priv_1 on a server under her control (e.g. her work station that runs a daemon process while she is absent) and gives priv_2 away to Bob. Observe that for this scenario, a distributed key generation is not necessary.

Incoming mail encrypted with pub is processed as follows. After having reached the server, the email is *partially* decrypted with priv_1 and forwarded to Bob (possibly after having checked some constraints such as sender or the current date). Partial decryption means that its content is still completely unintelligible to third parties, but may be decrypted by Bob. This is similar to a partial signature, which is no valid signature with respect to the public key, but can be made one with the aid of the second party.

As a PSE, Bob is given a standard PKCS#12 file by Alice at setup, which he can install in his MUA. This is the only thing he has to do, in particular

there is no additional software necessary on Bob's machine. At any point in time, Alice has the situation well in hand as she can always deactivate the forwarding entry on the server (or delete the share priv_1 completely) and Bob will learn nothing about future encrypted messages.

We briefly sketch the cryptographic details. Note that hybrid encryption is used in this setting. Threshold cryptography computations are only done on the encrypted ephemeral session key, not the symmetrically encrypted message itself. This means that m in Figure 5.14 and the following has to be interpreted as the session key. For RSA, where $\langle \mathsf{priv}, \mathsf{pub} \rangle = \langle d, (N, e) \rangle$, we use a multiplicative sharing $\mathsf{priv}_1 \cdot \mathsf{priv}_2 \equiv \mathsf{priv} \mod \varphi(N)$ as this allows seamless integration on Bob's side. Here the partial decrypt functions are identical to standard RSA decryption: $\texttt{partialdecrypt}^{(i)}_{\mathsf{priv}_i} = x \mapsto x^{\mathsf{priv}_i} \mod N$.

Threshold decryption with seamless integration at the proxy's side also works with the ElGamal cryptosystem. ElGamal can be realized in any finite cyclic group $G = \langle g \rangle$ (written multiplicatively) where computing discrete logarithms to the base g is difficult (e.g. the multiplicative group of a prime field). Here priv is a secret exponent, which we share additively modulo the order of g: $\mathsf{priv} \equiv \mathsf{priv}_1 + \mathsf{priv}_2$. The public key pub consists of the description of the group G, its generator g, and g^{priv}. Encryption of a message $m \in G$ is done via picking a random integer k and computing the tuple $c := (g^k, (g^{\mathsf{priv}})^k m)$. Here, $\texttt{partialdecrypt}^{(1)}_{\mathsf{priv}_1}(c) = (g^k, (g^{k\mathsf{priv}_1})^{-1} m)$ and $\texttt{partialdecrypt}^{(2)}_{\mathsf{priv}_2}((g^k, \mu_1)) = (g^{k\mathsf{priv}_2})^{-1} \mu_1 = m$. Note that $\texttt{partialdecrypt}^{(2)}_{\mathsf{priv}_2}$ is the standard ElGamal decryption function.

5.5.2 Securing an Enterprise Gateway for Secure Email

Having recognized the need for usable secure email solutions, some software makers and open-source initiatives have come up with so-called enterprise gateways for secure email. These systems shift the burden for cryptographic operations from the end user's MUA to a centralized system connected to the mail server in the enterprise network or even directly integrated with it. The idea of crypto email gateways is described in detail in [209, 183]; Gerling and Kelm [101] give an overview over existing products.

These solutions all have in common that cryptographic operations are delegated to a central authority. For technical reasons, such an authority

requires a direct connection to the Internet and has to store all decryption and signing keys. This makes it vulnerable and attractive both to internal or external attacks. From the users' viewpoint, the gateway has to be trusted to a large extent since it has the capability to impersonate them or steal their decryption keys. As a practical consequence, the administrator of the corresponding computer has to be fully trusted or the system has to be made robust against tampering. The former is a quite unsatisfactory and often unrealistic condition whereas the latter is a complicated task since administrators (must) usually have extensive capabilities on the operating system level. To prevent unauthorized access to the central key store, either by a fraudulent administrator or an intruder, we propose to again enforce a four-eyes principle with the help of threshold cryptography. We have shown the feasibility of this approach by our prototype *SecMGW* (Secure Mail GateWay) which is described in [234] in more detail.

Figure 5.15 illustrates how threshold cryptography may works in the email gateway setting: An encrypted message is delivered to the company's mail server connected to the Internet. This server uses the share priv_1 (of the corresponding recipient's private key) to compute $\mathtt{partialdecrypt}^{(1)}_{\mathsf{priv}_1}(c)$ and forwards it to its counterpart. Note that this partially decrypted message is unintelligible to a potential eavesdropper on the internal network. When the recipient connects to her mailbox to download the message, the respective server applies $\mathtt{partialdecrypt}^{(2)}_{\mathsf{priv}_2}$ and returns the plaintext (via a secure channel). The partial decryption functions for RSA as well and ElGamal are realized in the same way as in the previous section.

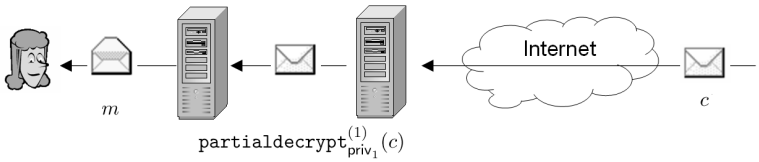

Figure 5.15: Decrypting email using threshold cryptography.

It has be shown above that a single share does not provide any information about the private key or any advantage in order to decrypt the ciphertext. If all end user keys are shared between two computers (that are administered by different persons) according to Figure 5.15, even compromising one subsystem would not harm overall security. The extra costs

for implementing this four-eyes principle are tolerable since both systems operate in nearly the same way. Initial key generation may take place on the client at the moment when a new user registers or even in a distributed manner with the two subsystems involved.

We emphasize that the four-eyes principle can be generalized to a k-out-of-n secret sharing. This means that each private key is split into n pieces requiring the cooperation of an arbitrary subset of k shareholders for each private key operation. For instance, 2-out-of-n schemes can be used for load-balancing since cryptographic computations are time-consuming.

5.6 Conclusions

An important way to manage the complexity of security in general and PKI in particular is to outsource critical and complex tasks. In this chapter we looked at the problem of PKI outsourcing and investigated a common certificate enrolment scheme. An important weakness was found, which can be exploited by a malicious service provider to impersonate its customer. We consider this an intolerable risk, especially as one concrete incident already has become public and has underlined the severity of the issue. Fortunately, threshold cryptography provides an elegant, yet provably secure way to solve the problem. We argued that shared digital signatures also require a distributed generation of the certification key. This led us to an open cryptographic problem, namely the efficient generation of a distributed RSA key by two parties. One main part of this chapter copes with this problem. Fortunately, we could design such a distributed protocol for two parties with the help of multi-prime RSA. Its complexity is only linear in the bit length of the modulus compared to the quadratic complexity of previously known schemes, while its security can be shown under the same standard assumptions. We provide a software package for distributed certificate signatures and DSA key generation as well as a prototype for RSA key generation. Our experiments have shown that the costs for two-party shared RSA key generation with the new algorithm are no more prohibitive.

Our proposal for a new enrolment protocol is flexible in many respects. First of all, there are up to three integration variants of the new scheme. One of them allows to transparently retrofit the mechanism into an existing trustcenter without changing anything at the CA. Furthermore our protocol

works well with every signature scheme that permits a shared generation of a digital signature. Examples include ElGamal, DSA, and RSA. We saw that discrete log-based signatures schemes are superior to RSA with respect to the costs of key generation, but they do not allow retrofitting our new enrolment scheme so easily to an existing process. Another degree of freedom is the treatment of multiple RAs, which can be integrated in a flat or two-stage hierarchy. The idea of shared signatures can also be used for the signing of CRLs and OCSP responses, although it is not so security-critical to shift this task completely to the service provider.

As an outlook we presented two more applications of threshold cryptography to email scenarios. One deals with the security of private keys on an enterprise-wide crypto gateway. The other solves the problem of how to give a vacation replacement the power to temporarily read one's encrypted mail without handing over the private key. We expect to see more applications using threshold cryptography in the future.

Chapter 6

Opportunistic Security for Email

The vast majority of users detest anything they must configure and tweak. Any really mass-appeal tool must allow an essentially transparent functionality as default behaviour; anything else will necessarily have limited adoption.

– Bo Leuf

We're not designing a system to handle nuclear weapons launch codes.

– Peter Gutmann

6.1 Introduction

In this chapter we present a novel approach to secure email which is based on the idea of opportunistic, i.e. best effort, security. The poor user acceptance of secure email led us to the conclusion that a trade-off between security and usability must be made in favour of usability. This may imply a level of security that is less than perfect, but which is still "good enough" with respect to a plausible threat model (cf. the discussion in Section 2.3.2).

It is common knowledge that the amount of secure messages is only a negligible fraction of the total amount of email. Obviously, there are two reasonable explanations for the lack of sufficient incentives for securing mail: First, the *anticipated benefits* of secure mail are too low, or second, the *costs* of applying the existing security technology are too high.

The benefits appear to be low if the threat perception is weak, which is generally the case as we argued in Section 4.2.1. Consequently, the costs of security must be low as well for security to be attractive. The costs can be measured in monetary terms or in terms of the overhead users have due to security technology. Monetary costs can be discounted since numerous PKI-enabled email clients are available and certificates can be obtained for free or at affordable prices (see Sections 3.4.3 and 3.2.4). The costs are thus determined primarily by the *cognitive effort required to operate electronic mail security.* This implies that users install the necessary software and obtaining key pairs and certificates. As we stated in Chapter 3, the difficulty lies in exchanging keys, building trust in the received keys, authorizing cryptographic operations, and in diligently and competently reacting to prompts and warnings of the security software.

The research question then becomes what the optimum trade-off should be and how the security benefits are maximized with minimum damage to usability. Based on our analysis in Section 3.2, we argue that public key certificates signed by intermediaries (i.e. CAs or PGP peers) and digital signatures account for a considerable cognitive or operational overhead, but they contribute marginally to the practical level of security. By eliminating these primitives, the costs versus benefits ratio can be improved. Towards more user-friendly secure mail, we explore a non-intrusive approach that works without either of these primitives, and rather focuses on transparent message encryption and integrity protection. In our approach, we separate key *exchange* from *binding keys to identities.*

This chapter is based on our paper [193]. It starts with a summary of the threat model and an analysis of what suitable trade-offs between usability and security might be. The conclusions we draw from our analysis guides the design choices we make in subsequent sections. In Section 6.3 we describe the processes of key management and key rollover, before we briefly explain the technical aspects of protecting mail. The new user interface design and new interaction mechanisms for the verification of keys are the subjects of Section 6.5. We conducted two user studies to assess the usability of new metaphors and the complementary visualizations of the security state. The results are given in Sections 6.5 and 6.6. The latter one provides evidence that pairwise manual key fingerprint verification can be feasible in everyday

usage. This chapter ends with a discussion of related work and a conclusion and outlook to further work.

6.2 Background

6.2.1 Email Threat Model

We focus on (non-commercial) individual users of email who communicate by mail from their homes. We call them again Alice and Bob. Both connect to the Internet through a telecommunications provider ("Telco") and an ISP. Alice sends her mail by transacting with a mail exchanger (MX), for instance a *sendmail* daemon run by her mail provider. En route to Bob, her mail may be forwarded several times from one MX to another until it is delivered to Bob's mail drop. The mail drop stores messages for Bob until he retrieves them with his email client by means of the POP3 or IMAP protocols or a web front end. Figure 6.1 illustrates that model. Here the physical and logical channels (dotted lines) of the email are depicted in a single graph (cf. Section 2.1.1). We assume that attacks can occur at any entity or connection channel except at the PCs of Alice and Bob.

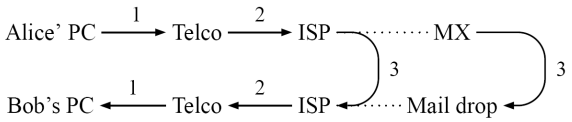

Figure 6.1: Physical and logical path of an email.

We distinguish between three classic categories of threats as suggested by Anderson [9]. Those threats correspond to the security goals of confidentiality, integrity/authenticity, and availability.

Unauthorized information release: The adversary, Mallory, intercepts and reads mail sent by Alice to Bob. We do not address the threat of traffic analysis, that is beyond our scope.

Unauthorized information modification: Mallory modifies email sent from Alice to Bob while it is in transit or in storage, or she sends mail to Bob that is forged in the name of Alice.

Unauthorized denial of use: Mallory can prevent Alice from sending mail to Bob, or can prevent Bob from receiving mail from Alice. Note that

a powerful adversary can always deny service by saturating or severing the communication channels. We limit ourselves to denial of service through regular interaction with Alice's and Bob's mail system, e.g. by sending them forged or unsolicited mail.

Mallory may or may not collaborate with any of the entities involved in forwarding or storing mail with their consent (e.g. implied by search warrant) or without their consent (e.g. because Mallory compromised a MX). Mallory may launch passive attacks (eavesdropping only) or active attacks, which involve the modification or generation of messages. Although we discuss threats and countermeasures in terms of Alice and Bob, which is a particular pair or users, our intention is to address large scale attacks. In other words, Mallory attacks multiple mail "sinks" for a mail "source," multiple mail sources for a mail sink, or multiple sources and sinks.

6.2.2 Potential Tradeoffs

Let us briefly recall a few prudent engineering rules for designing protection mechanisms due to Saltzer and Schroeder [199], which we have already introduced in Section 2.3.4.

Economy of mechanism: The design should be as simple and small as possible.

Fail-safe defaults: The mechanism should have fail-safe defaults.

Psychological acceptability: The mechanism should be easy to understand and use.

From an economic or rational perspective, public key certificates are worthwhile if their benefits outweigh their costs. The primary benefit of certificates is the prevention of impersonation attacks by *active* adversaries who substitute their keys for the keys of legitimate users. For comparison, public key cryptography without certificates foils *passive* eavesdroppers completely [74], and it uncovers active adversaries unless the adversary intercepts and re-encodes all mail *at all times,* particularly at the time when the communicants first exchange their keys.

In order to fulfil their objective, certificates require a complicated PKI. We argued that certificate validation and key management accounts for

the majority of the interactions and decisions that interfere with the goal-oriented tasks of a user, and which the user has difficulties to understand. Since a PKI seems to be neither simple (first rule) nor particularly easy to understand and use (third rule), we decided to investigate what it would take to achieve adequate security without that primitive. Our objective is not to attain 100% security but to attain significantly improved security at a high degree of usability.

Separating Concerns Our first measure was to separate concerns that would be uniquely addressed by a PKI, which is the process of binding a key to an identity. Once that was determined, other separations of concerns followed suit. (These issues are addressed in the four following sections.)

1. exchanging keys and maintaining current keys
2. protecting mail
3. communicating and responding to potential security breaches
4. *optionally* binding keys to an identity

Here, "binding" implies verification of an ad-hoc association between a key and an alleged identity. For as long as the user is not concerned about active attacks, she may disregard the binding and trust the ad-hoc association. This does not necessarily mean that the association is not trustworthy, each concern has elements that attempt to detect security breaches based on certain *coherence principles* that we define along with our mechanisms. For instance, if Alice uses public key k in her communication with Bob then she may not use public key $k' \neq k$ unless she introduced k' and proved that she knows the private keys that belong both to k and k'. A variety of more general coherence principles have been summarized nicely by Arkko and Nikander [14] under the notion of *weak authentication*.

Design Objectives In summary, our approach is to make key exchange and encryption a largely *transparent default behavior* and to design protocols that make it difficult for Mallory:

1. to initially compromise a communication channel,
2. to maintain continuous control over the channel,
3. to leave the compromised channel undetected,

even if no binding between keys and identities is provided through a PKI. Thereby, we aim to reduce the overhead (costs) associated with the operation of mail security to a degree where mail security becomes ubiquitous. At the same time, we raise the risks and costs of successful attacks so that a rational adversary must concentrate her resources on a small number of targets with a high overall yield – and thus does not invade the privacy of individual users on a large scale.

6.3 Exchanging and Maintaining Current Keys

6.3.1 Key Exchange Process

Our first concern is to distribute public keys that can be used to prevent unauthorized information release or modification. At this point, we are *not* concerned about the binding of a key to an identity, we cover that aspect further below in Section 6.6. That enables us to design the exchange of public keys in a fashion that is entirely transparent to users, unless of course there is evidence of a potential attack. The (fully automatic) key exchange process of Alice ("Kex") has three duties:

1. maintain the "security state" of Alice, which is the known set of keys, peers, and their relationships;
2. decide which keys must be used and which keys must be exchanged at any given point in time;
3. update the security state based upon the keys that are received.

6.3.2 Transformations of the Security State

The principal idea is to design Kex such that updates of the security state, in other words a "transition" in the state space, occur only if the new state is at least as secure as the previous state. A transition is triggered if, as part of a mail, a new key is "introduced," which requires a message from Bob to Alice of the following form:[1]

$$\text{Bob} \rightarrow \text{Alice}: \ \{\text{“Bob”}, \text{“Alice”}, t, h(k_{\text{old}}), k_{\text{new}}\} \text{ signed with } k_{\text{old}}^{-1}, k_{\text{new}}^{-1}$$

[1] For simplicity, we assume that the same key pair is used for signing and encryption, as is common practice in S/MIME-based applications. The approach can readily be extended to support dual key pairs.

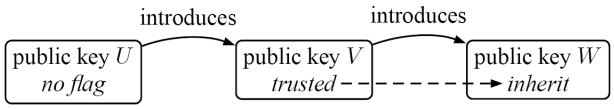

Figure 6.2: Schematic view of key structure.

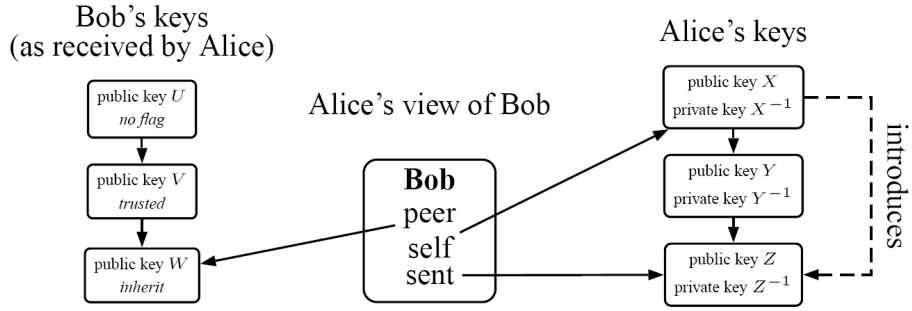

Figure 6.3: Data structure Alice keeps for each of her peers.

Here, t is the current time, h denotes a cryptographic hash function and $\langle k^{-1}, k\rangle$ is Bob's key pair. In practice, "Alice" and "Bob" would denote Alice's and Bob's canonical email addresses. The data structure carries a double signature (cf. Section 5.2.7) produced by the new and the old private key.

The semantics is that Bob tells Alice that he wishes to replace his older public key k_{old} with the more recent public key k_{new}. As a shorthand, we write $k_{\text{old}} \succ k_{\text{new}}$, provided that both signatures are valid. The first key is introduced self-signed, i.e. k_{old} is empty. Old keys and the order in which they were introduced are kept (see Figure 6.2) until they are not referenced any longer in a peer structure and are garbage collected (for instance based on expiry time and storage quotas). Key structures are kept in linked lists. Each successor key k_{new} is introduced by its predecessor k_{old}.[2] A *trusted* flag indicates that the key has been successfully bound to an identity. The flag carries forward, which means that a successor key with no flag set inherits the flag settings from its predecessor.

[2]With respect to the signature of public keys, key structures are comparable to a certificate chain where each certificate contains a proof-of-possession for the corresponding private key.

Data Representation Kex intercepts incoming mail and processes any valid key introductions contained therein using three types of data that together comprise the security state (see Figure 6.3):

1. The list of Alice's key pairs ordered by recency.
2. For each peer (e.g. Bob) his associated public keys ordered by recency.
3. A peer structure for each peer that references current keys.

Peer structures are persistent and created on demand when the first email of a peer is received. For instance, the peer structure that Kex keeps on Bob stores the following information:

self key: the most recent key of Alice that Bob used to protect his mail to Alice;
sent key: the most recent key of Alice the Kex introduced (sent) to Bob;
peer key: the most recent key of Bob known to Kex;

Processing Rules When Kex receives $k \succ k'$ it enforces a simple rule. If no key is known for Bob then a self-signed key is accepted as Bob's initial key and an association with Bob's identity is formed. Otherwise, if k is the known key of Bob then that key introduces k' which becomes the new most recent key of Bob. Of course, the introduced key k' must not already exist in Alice's security state or else an error occurs (that prevents the inadvertent or adverse introduction of loops in the history of keys). If an error occurs then Kex tags the mail that contained the key introduction with a descriptive error code.

Kex also intercepts outbound mail and determines for each recipient whether a key must be introduced (e.g., if Bob has not yet used any of Alice's keys or uses a key that is outdated). Once Kex selected a key for introduction, it keeps introducing that key with the most recent key of Alice that Bob knows until Bob begins using the introduced key. At this point, Kex either stops adding key introductions to outbound mail or starts introducing a more recent key of Alice (if such a key has been generated meanwhile).

Figures 6.4 and 6.5 give a more precise and formal presentation of how Kex processes incoming and outgoing key introductions. Note that in a self-signed key introduction the old key k is nil ("not in list"). If no peer key is known then *peer* is also nil (compare Figure 6.3). This allows a very

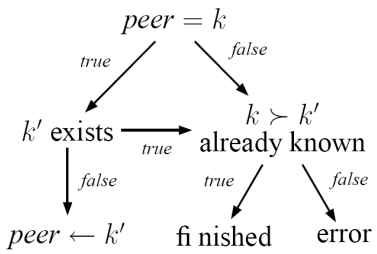

Figure 6.4: Decision graph for incoming key introductions $k \succ k'$.

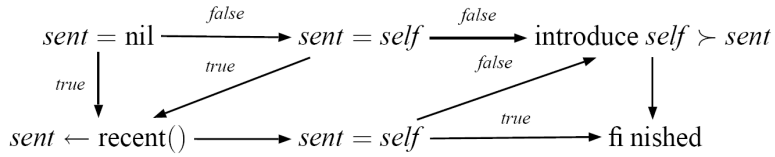

Figure 6.5: Decision graph for outgoing mails.

compact presentation of the graph. The *peer*, *self*, and *sent* variables in Figure 6.5 refer to the peer data structure that we introduced in Figure 6.3. As an invariant, the *sent* key must always be at least as recent as the *self* key.

The processes are designed to keep Alice's and Bob's security states synchronized even if an active adversary drops legitimate emails with key introductions, or reorders such mail.

6.3.3 Security Considerations

Obviously, a valid key introduction can be computed only by someone who knows the private keys that correspond to both public keys, the old and the new one. Alice does not really know whether Bob or Mallory sent the key introduction, Mallory could have generated both the old and the new key. However, by simple induction we can then reason that Alice must have received the first self-signed key with Bob's name on it from Mallory (we disregard the possibility that Bob adversely or inadvertently disclosed his private key to Mallory or that Mallory has broken Bob's key pair). On the other hand, if Alice received the initial key introduction actually from Bob (whether or not Alice is certain about it) then Mallory cannot send valid key introductions in Bob's name unless she breaks Bob's key pair.

Assume now that Mallory sent the initial key introduction. Alice attempts to send mail to Bob in Mallory's key which triggers alarms on Bob's side unless Mallory performs a MITM attack and transcodes all such mail. A single missed mail reveals Mallory's presence. Once Mallory starts her attack he cannot leave the communication channel between Alice and Bob without his presence being noticed. In order to do so, she would have to introduce Bob's public key to Alice which requires forgery of Bob's signature. Alice's advantage is that Murphy's Law is in her favour – eventually Mallory will make a mistake, perhaps due to a configuration error, a route change, a computer failure, or because Alice and Bob followed procedures we outline in Section 6.6.

Finally, assume that Mallory broke Bob's key. That means she can read Bob's mail to Alice – until Bob introduces a new key. At this point, security is either restored or Mallory must launch an active attack with all the implications discussed above.

In summary, we presented a secure key exchange and key maintenance scheme that operates *continuously* and *transparently* on Alice's inbound and outbound mail. No user interaction is required unless evidence of an attack is found. Powerful active adversaries may still attack but their chances of evading eventual detection are limited.

6.4 Protecting Mail

Email consists of headers and the mail body (RFC 822, see [58]). Both elements need protection, which is achieved by transforming them into a separate *envelope* for each intended recipient. The envelope consists of a minimal set of headers and a protected body, and is sent to the recipient (e.g. Bob) via the simple mail transport protocol (SMTP). Note that the confidentiality or authenticity of email header fields like the sender or subject is not protected by S/MIME. How exactly the envelope is represented is largely a matter of implementation choice. On an abstract level, the process is quite simple. If Bob requires a key introduction then it is added to the mail. Key introductions, headers, and mail body are jointly signed with the introduced key (the *sent* key) or otherwise with the most recent key of Alice that is available to Bob (the *self* key).

6.4.1 Outgoing Email

Finally, if a public key of Bob is known ($peer \neq \text{nil}$) then the signed information including signature is encrypted with that key and the cipher text is placed in the body of the envelope. The information necessary for decryption (e.g. key identifiers and encrypted content encryption keys) is added where appropriate as per implementation choice. Necessary and sufficient information to fulfil mail transport requirements are copied to the envelope's header section.

If no suitable key of Bob is known then the augmented email (and not the envelope) is sent (unencrypted) or not sent at all based on a prior policy decision that we explain in greater detail in Section 6.5 along with user interaction principles.

6.4.2 Incoming Email

Inbound mail is processed in reverse order, which is a more involved process due to the various error conditions that may arise. For simplicity, let "Kin" be a shorthand for that process. If all applicable verification operations turn out correctly and the encryption key is more recent than the key previously used by Bob, then Kin updates the *self* key in Alice's view of Bob accordingly. In case of errors, Kin signals them by adding appropriate headers to the email, which must then be interpreted by the MUA. In addition to rather obvious error conditions such as invalid signatures and missing keys, there are some subtle conditions that Kin must verify. For instance, if Bob once sent mail to Alice which was signed or encrypted, then any subsequent mail without that protection is flagged with an error. Otherwise, adversaries could simply bypass the security mechanisms by sending plain text mail, which is easily forged.

6.5 Key Introduction and User Interaction

Up to this point, our discussion focused on the key exchange and mail protection processes that underpin our engineering approach. Most importantly, we separated the key exchange from binding keys to identities (in a strong sense) and we made transparent signatures and encryption the default mode of operation. In this section, we discuss how that approach enables us to simplify the interaction with users.

The usability of contemporary mail security software is limited primarily by two factors:

1. required interactions with security functions are perceived as an obstacle to accomplishing goal-oriented tasks (see e.g. [204]);
2. users have limited understanding of the concepts that underpin trust models and cryptographic primitives (compare e.g. [60, 252]).

We illustrate these limitations with two examples. First, warning messages and confirmation boxes frequently appear and must be "clicked away" when no suitable keys are available even though the user may be well aware of the risks of sending mail in the clear and is willing to accept them. Second, users are presented unintelligible information, for instance, certificate contents which are easily set to arbitrary and superficially convincing values by adversaries who generate self-signed certificates.

In contrast, our user interface design concepts require fewer interactions than contemporary approaches. We let users interact with concepts they know ("mail" rather than "keys"). Our objective is to provide a familiar context and mental representation [179]. All decisions that a user must make are related to a purpose (e.g. confidentiality) rather than unfamiliar concepts (e.g. certificates as the means to an end). Furthermore, users are not forced to make decisions immediately, they are at leisure to invoke security functions when deemed appropriate, which is influenced by users' curiosity or risk perception. We cover two user interaction aspects in this section: output (i.e. alerts and visualizing status information such as the availability of a particular key) and input (i.e. making policy decisions).

6.5.1 Alerts and Status Display

Alice must be alerted if the key exchange and maintenance processes detected incoherent key usage, invalid signatures, or other error conditions. Such alerts are associated with a particular mail that has been received. A simple and non-intrusive method to alert Alice to such an event is to highlight the suspicious mail prominently, e.g. in reverse, in the mail display (see Figure 6.6). For example, assume that Bob introduced his key to Alice in one mail and Mallory subsequently sends a key introduction of her own forged key in Bob's name. However, Kex would flag Mallory's mail with an error and the mail would be highlighted to bring the fact to Alice's atten-

tion. On the other hand, if Mallory sent her mail before Bob then Bob's mail – the genuine one – would be flagged. In either case would Alice be alerted to the fact that some sort of attack is ongoing. Which key is which – and consequently which email is genuine – can be determined by appropriate subsequent verification. However, it is upon Alice to decide whether or not to investigate the alert, and she may do so at a time of her choice. Her regular tasks remain largely uninterrupted.

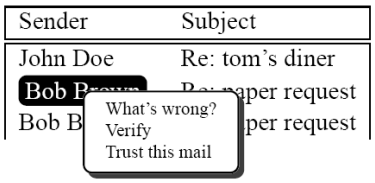

Figure 6.6: Context menu of a mail flagged with an error.

Alice may investigate the reason for the alert and the conclusions that can be drawn from it by invoking a context menu with a right click. The context menu allows to inquire about the reason of the error, to verify the association of a key and an identity, or to define the underlying key as trusted (which binds the key to the identity of the sender). These alternatives are shown in Figure 6.6. Subsequently, she may wish to invoke a verification procedure (see Section 6.6) which allows her to confirm the authenticity of the mail. It is worth noting that technically the authenticity of the *key* would be verified that the sender of the highlighted mail used. However, Alice would have the illusion that the verification refers to the mail rather than the keys. Based on the outcome of the verification procedure, Alice may define the mail (or rather the underlying key) as trusted. Once a key of Bob is trusted to be authentic (and marked appropriately in Alice's security state) that trust extends to all keys that Bob introduces with the trusted key (see Figure 6.2 again).

6.5.2 Input of Policy Decisions

If Bob's key is unavailable to Alice at the time when Alice wishes to send him mail then current MUAs typically alert Alice by raising an alert panel. The dialoge then requires that Alice confirms or cancels the send operation by a mouse click. That forces Alice to divert her attention to handling the interruption before she can proceed with her original work process. If

Alice is an inexperienced user, then she may even be forced to switch input modalities (e.g., from keyboard to mouse) in order to handle the event. It is less intrusive to communicate the alert by highlighting Bob's mail address appropriately while Alice enters it. For illustration, see Figure 6.7: Bob's address bob@brown.net is displayed in an emphasized way (e.g. in colour and/or reverse video) to indicate that his key is unavailable. The entry of the second address is still in progress, so no highlighting has taken place yet. After entering multiple recipient's addresses and the email body, Alice may still decide what to do with Bob's address (or to ignore the event) without a major interruption of her task. She may chose to keep it or delete it from the recipient's list.

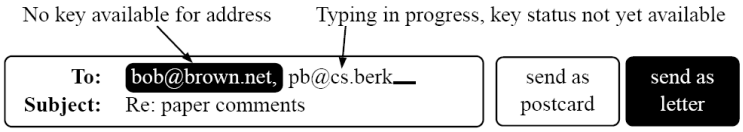

Figure 6.7: Feedback on key status and policy options while typing.

Use of Metaphors for Secure/Insecure Email Additionally, we consider analogies to the physical world when requiring input of policy decisions, i.e. which alternative action is to be taken when no key is available for a mail's intended recipients. In the physical world, we send postcards or letters. Postcards are readable by anyone involved in the delivery process and consequently the expectation of confidentiality is low when sending them. Letters have been opened regularly (and re-sealed imperceptibly) for reasons of espionage during the Second World War [135], and there is no reason to believe that letters are not intercepted and read today. However, for all practical purposes, letters offer a much higher degree of confidentiality, and consequently the users' expectations of confidentiality are higher as well. Electronic mail sent in clear text resembles postcards whereas encrypted mail resembles letters.

Consequentially, we substitute the common "Send" button by two buttons: a "Send as postcard" and a "Send as letter" button (see also Figure 6.7). The former encrypts mail for all recipients whose keys are known and sends clear text to all other recipients, whereas the latter sends encrypted mail to those recipients whose keys are known and no mail to all

other recipients. In either case, mail is encrypted automatically whenever a key is available. In order to prevent users from inadvertently not sending mail to a particular recipient whose key is unknown, the "Send as letter" button is highlighted if the key of one or more recipients is unknown. As in the case of handling alerts, this approach conveys all necessary information to Alice without necessarily demanding additional user interactions. Sending an email still only requires one click and no alert boxes interrupt Alice's work task subsequent to sending a mail – all security policy decisions are made beforehand.

6.5.3 Evaluation of Metaphors

We ran a preliminary user study to determine how well subjects intuitively understand our letter and postcard metaphors. Towards that end, we conducted a sequentially structured interview with 5 female and 14 male subjects of age between 20 and 49 years (average age: 29 years). They were given a colour printout of two mock-up user interfaces (based on the Mozilla Thunderbird client), one with labels and the other without them. The version with labels is shown in Figure 6.8. The captions are given in German, their meaning is "send as letter" and "send as postcard". The interview was structured as follows:

- Subjects were shown the mock-up UI without button labels and were asked to guess the functionality of the icons.
- If subjects did not correctly guess the functionality, then they were shown the same mock-up UI with labels and asked again.
- If subjects did not correctly guess the functionality, then they were told that the UI belonged to a new mail client with improved security functions. Subsequently, the subjects were again asked to guess the functionality of the icons.

Subsequent to six test runs, we also began to collect demographic information, and assessed subjects' mail handling expertise, threat perception in general and with respect to the risk of eavesdropping, and willingness to accept higher personal efforts to secure their mail communication (so only 13 results are available for these questions). The corresponding results are shown in Figure 6.9.

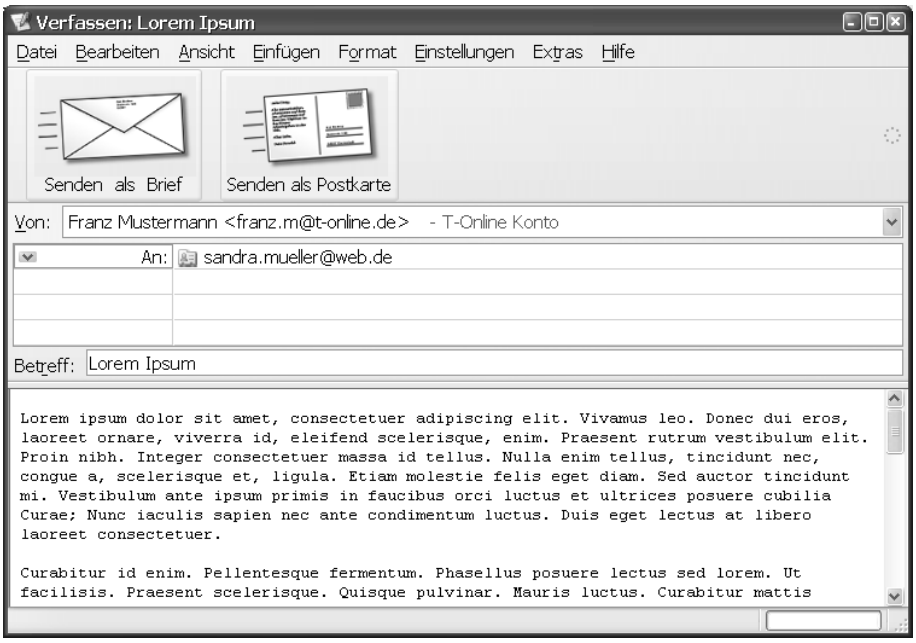

Figure 6.8: Paper mock-up used to evaluate letter/postcard metaphors.

All subjects recognized the icons as what they were. All but two subjects understood the metaphors, six without labels, three after icon labels were shown, and eight after the context hint was given. The metaphors chosen were rated as "good" even though some participants mentioned that the symbols could be enhanced by "common security metaphors" such as a padlock or a seal on the letter. Incorrect interpretations concentrated on the postcard. Subjects interpreted that icon as "send summary" or "send multimedia". We found no correlation between correct recognition and demographics or experience. However, we found a trend in the correlation between understanding the metaphors and a high threat awareness ($r = 0.43, p < 0.07$), as well as between understanding of the metaphors and willingness to accept a larger effort for securing mail ($r = 0.36, p < 0.09$).

The results of our study give us reason to believe that our metaphors can be adopted without significant learning effort. The "fine print" of the security policies conveyed by our metaphors can be explained to the user by means of a pop-up hint that should be shown for instance upon first invocation of the new functionality (perhaps with a "don't show again" check mark). Note that the policy is "implicit" in the metaphor – users do not define a custom policy to be associated with the metaphors. Additional

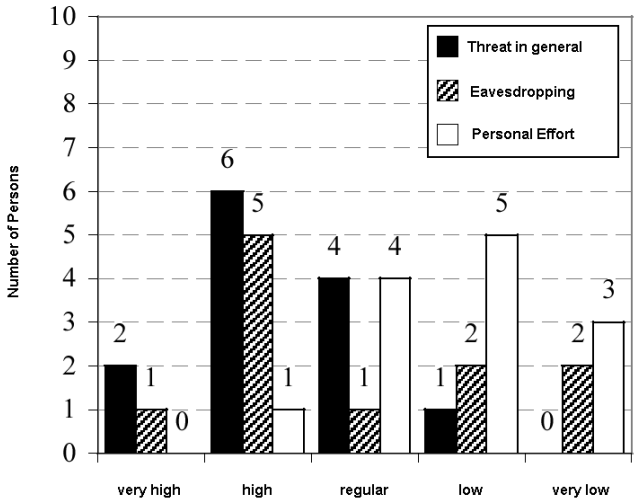

Figure 6.9: Interview results metaphor evaluation.

feedback mechanisms are conceivable in the "send letter" case if the keys of some intended recipients are not available. For instance, an idea might be to offer the sender the possibility to split the email in two messages, one addressed to the recipients who are able to receive confidential email and the other one for the remaining peers. If the email contains sensitive pieces of information, the sender has now the possibility to exclude all or some of them in the insecure mail.

6.6 Binding Keys to Identities

Above, we described how mail can be protected against unauthorized information release and modification in a fashion that is largely transparent and non-intrusive. Although, we have not obtained absolute security – a determined adversary who manages to intercept the first key exchange between Alice and Bob may launch a MITM attack against them, and may even evade detection for an unspecified period of time. The reason is that our approach does not, in a strong sense comparable to a public key certificate, bind a key to the identity of its legitimate owner. We expect that Alice and Bob achieve such a strong binding by verifying their keys over an authenticated channel.

A legitimate question one may ask is whether that approach is a scalable and what burden is placed upon users by it. Below, we present results of an

analysis of users' mail behavior that we conducted to find an answer to said question. More precisely, we analyzed the *setup costs* and the *continuous costs* of pair-wise key verification:

1. the setup costs are measured by the size of the peer group that a subject has when the system is introduced;
2. the continuous costs are measured by the rate at which the subject encounters new peers.

Our results, which we summarize and interpret below, led us to believe that pair-wise key verification may indeed be scalable. Furthermore, the analysis leads to a *greedy* algorithm which maximizes the security benefit per key verification so that users can determine an optimal trade-off between their desired level of security and interaction overhead.

6.6.1 Materials and Methods

We developed a set of tools (using Java, Perl, MatLab, gnuplot, and Bourne Shell) to analyse mail stored in IMAP or POP3 folders, and file system-based mail drops in the mbox or related formats (allowing us to access mails stored in Mozilla, Thunderbird, Opera Mail, Pegaus, and – via an additional converter – Outlook Express). Our scanner extracts the *To, Cc, Bcc, Reply-To, Date, References,* and *In-Reply-To* headers of all mails. Electronic mail addresses are anonymized by substituting them with a running identity number ("ID") that is the same for equal addresses. Equality of email addresses is determined by a lower-case comparison of their local part and domain [58].

Each ID stands for a potential *peer* of the mail owner (for ease of description we refer to the mail owner as Alice). A peer of Alice is a communicant with whom Alice has a bidirectional communication, i.e. Alice sent mail to her peer and the peer sent mail to Alice (or vice versa). Our tools determine the set of Alice's peers according to the following rules:

1. Alice provides her own addresses (one or many)
2. A mail is an answer if it has a References or In-Reply-To header
3. A sender is a communicant who sent mail to Alice
   (sender $\in$ From $\cup$ Reply-To and Alice $\in$ To $\cup$ Cc $\cup$ Bcc)
4. A *strong* sender is a communicant who replied to Alice
   (sender $\in$ From $\cup$ Reply-To and Alice $\in$ To and the mail is an answer)

5. A receiver is a communicant to whom Alice sent mail
   (Alice $\in$ From $\cup$ Reply-To and receiver $\in$ To $\cup$ Cc $\cup$ Bcc)
6. A *strong* receiver is a communicant to whom Alice replied
   (Alice $\in$ From$\cup$Reply-To and receiver $\in$ To and the mail is an answer)
7. A peer is a strong sender, a strong receiver, or in sender $\cap$ receiver
8. A peer is **not** a mailing list e.g., an address that includes:
   majordomo@, listserv@, -request@, -list@, etc.

For each mail and each sender and receiver referenced therein, our tool generates a sample which consists of the communicant's ID and the time stamp of the mail that is extracted from its Date header. Next, double entries are removed which may occur e.g., if mail is filed multiple times. Finally, we eliminate all samples that are not peers and all samples with an age of more than two years measured from the most recent sample.

Three cumulative distribution functions (CDF) are computed from the remaining set of samples:

1. percentage of mail exchanged with $n \leq x$ peers;
2. percentage of weeks where mail was exchanged with $n \leq x$ peers;
3. percentage of weeks where $n \leq x$ new peers were observed.

Multiple sets of samples (one set per mailbox owner) are combined by averaging the values of the distributions. The output are the mean $\bar{y}$ for each position x and the standard deviations for samples above (s_a) and below (s_b) the mean. In the Figures 6.11 to 6.13 the results are plotted as: $f_a(x) = \bar{y}(x) + s_a(x)$, $f(x) = \bar{y}(x)$, and $f_b(x) = \bar{y}(x) - s_b(x)$.

6.6.2 Description of Study

We generated a Web page from which our mail drop scanning tool and instructions for its installation and configuration (both in English and in German) could be downloaded. The start window of the statistics tool is shown in Figure 6.10. Here, participants could provide the location of their local and remote email folders. The subjects we invited to participate in our study were colleagues, fellow researchers, and students at Technical University Darmstadt, Fraunhofer Institute for Computer Graphics (Darmstadt), and the Peter Kiewit Institute at the University of Nebraska at Omaha. Overall 34 individuals responded by sending us their anonymized mailbox extracts. The sizes of the sample sets varied considerably as some participants

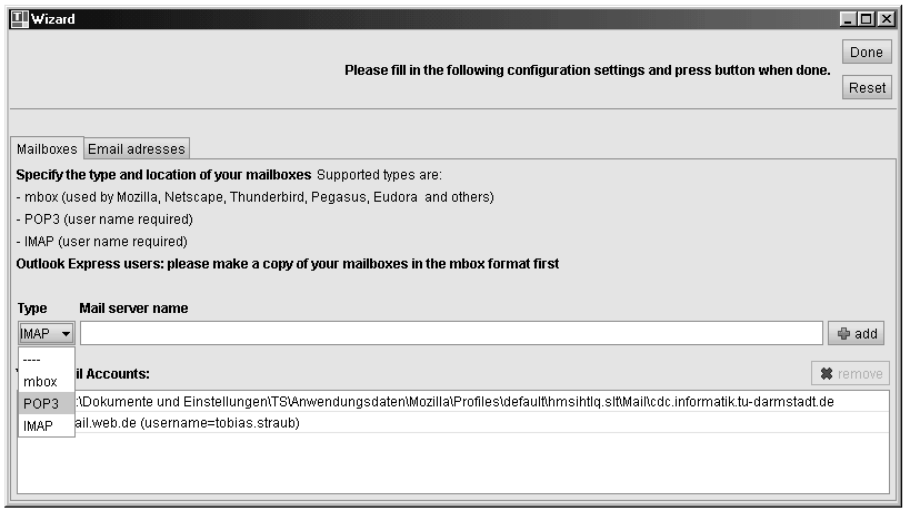

Figure 6.10: Start window of mailbox analysing tool.

were communicating significantly more by mail than others. For instance, the participant who provided the largest dataset is a system administrator responsible for an IT infrastructure that serves about 200 researchers, students, and IT personnel at spin-off companies. Other participants are actively engaged in collaborative research and project work.

We sorted the datasets by the number of samples our tool extracted from each set and chose the 17 sets with the largest numbers of samples, expecting to obtain more reliable results than would probably have been the case if the smaller datasets were included. All retained datasets had more than 750 samples after applying our filter procedure. Table 6.1 lists the sizes of the extracted samples ("all") and the samples that were retained after samples of non-peers were removed ("filtered"). The third column shows their quotient. The results of our analysis are displayed in Figures 6.11 to 6.13. The graphs show the CDF listed above, they answer the question, whether in y percent of all cases a given condition held for x or fewer peers. Each graph is plotted twice with different scales. On the left, the ordinate ranges from 0 to 1 and from 0 to 0.8 on the right. We found the following:

- On average 50% of all mail was exchanged with 10 or fewer peers ($\min(x \in \mathbb{N} \mid f(x) \geq 0.5) = 10$ in Figure 6.11), and with 21 or fewer peers in our "worst case" scenario ($\min(x \in \mathbb{N} \mid f_b(x) \geq 0.5) = 21$ in Figure 6.11).

filtered	all	ratio
2651	3624	0.73
18650	29695	0.63
2473	3926	0.63
2463	3225	0.76
874	1265	0.69

filtered	all	ratio
2072	3184	0.65
2766	3453	0.80
3359	3785	0.89
4031	4521	0.89
1604	2073	0.77
8349	14484	0.58

filtered	all	ratio
7657	13007	0.59
783	1727	0.45
5076	9721	0.52
1204	2259	0.53
8194	11431	0.72
2465	7171	0.34

Table 6.1: Numbers of extracted mailbox samples (attributed to bidirectional communication and total number).

- In 50% of all weeks, the number of peers with whom subjects exchanged mail was 10 or fewer on average and 22 or fewer in the worst case ($f_b(22) \approx 0.545$ in Figure 6.12).
- In 50% of all weeks, subjects encountered at most two peers on average ($f(1) \approx 0.47$ in Figure 6.13) and 3 or fewer in the worst case scenario ($f_b(3) \approx 0.54$ in Figure 6.13).

6.6.3 Interpretation of Results

Our analysis is based on subjects' saved mailboxes, which necessarily captures a limited view on their actual communication since they almost certainly have stored sent and received mail selectively. On the other hand, the type of subjects whose mail communication we analysed uses email more professionally and likely more extensively than private non-commercial users. It is probably fair to say that the results we obtained are an upper bound of what we could expect to see when analysing the mail communication of private non-commercial users, a view that has been confirmed by occasional informal interviews we conducted with such users. Judging by our results, it appears that:

1. a reasonably small number of key verifications would be required to achieve an "ideal" level of security for the majority of all exchanged mail;
2. a fairly small number of key verifications would be required on an ongoing basis to achieve an "ideal" level of security with new peers.

Also, our results must be interpreted in the light of the actual threat level. Users may not feel – nor have – the need to verify their keys with all of their

peers, which further reduces the costs of obtaining a level of security that is comparable to (or even exceeds that of) certificates issued by a certification authority.

6.6.4 Greedy Binding

The analysis we conducted based on subjects' saved mail can be implemented as an ongoing process e.g., in a MUA or in a mail security proxy. That process may then display on demand a ranked list of Alice's peers for whom a key verification would provide the maximum "utility". In other words, the process would suggest the peers with whom the largest amount of mail is being exchanged and with whom no key verification has yet taken place. The MUA may additionally visualize a security "score" (e.g. the percentage of mail that is exchanged securely) which may spur a special effort of the user to increase one's security so that the "high score" is reached (in analogy to playing a computer game).

Human-computer interaction support for key verification opens several avenues for further research and a wealth of alternatives can be explored. It is not our objective to give a complete solution in this paper, we rather wish to stimulate further research in this direction below.

Online real-time communication protocols [10, 139, 225] allow to reduce the amount of data for key fingerprint verification that has to be exchanged out of band. A reasonable level of security can be achieved, even when a short sequence of letters (comparable to a password) instead of the 20 or more letters a hash value consists of are transferred by the phone. Such protocols have been investigated primarily from the perspective of cryptographic strength. We argue that an investigation of usability issues of such protocols has merit. Clearly the security against MITM attacks is reduced, but not too much as we believe. Having an uncertainty level or "residual risk" between 2^{-32} and 2^{-16} – which is on the order of the annual probability of being struck by lightning or the likelihood of a processor bug during the computation – should be sufficient for everyday email of most users. In cases where the parties do suspect fraud, they may still chose to exchange the whole fingerprint.

Last but not least, the relative rate at which users exchange mail is not the only conceivable metric to optimize the ratio of interaction and security – other behavioural or information-theoretic metrics may be applicable.

6.7 Related Work

The idea of implicit security where user actions trigger security policy decisions based on the assumption that user intention conveys policy choices towards achieving a particular goal has already been presented in Section 2.3.4. We are in favour of this approach although it is of limited expressiveness in the email context where sending or not sending mail are the primary choices. In this sense, we already stretched the limits of implicit authorization by suggesting two alternative send buttons (metaphors of email transport) which both convey different security policies.

With respect to communicating a system's security state to users, Dourish and Redmiles [69] follow an approach that increases the amount of contextual information a user must cope with, in the expectation that this improves his intuitive understanding of the system. We, on the other hand, seek to minimize the distraction a user faces due to security overhead. Our approach bears some similarity to Whitten's staging approach (see Section 2.3.6). Both approaches give the user the freedom of choice when to improve her security, either by verifying the authenticity of keys or by progressing to the next stage. However, Whitten expects that through training the user eventually masters the concept of certificates, something that we avoid entirely. Garfinkel and Miller [93] conducted a user test based on a scenario similar to the one of Whitten [253]. They applied a simple colour-code scheme to indicate whether an email is signed by a key that is new, has been used before, or is different from the previous one. The results are promising as users of opportunistic email security were shown to be significantly less vulnerable to impersonation where a new key pair is used by the attacker.

The approach of Levien *et al.* [153] towards transparent Internet mail security requires a formalization of trust by certificates and explicit policies which is the opposite of what we aim to achieve. Our approach also differs from the one in [207], which accounts for public keys with a limited lifetime, but limits its discussion of key management to password disciplines. Arkko and Nikander [14] formalize the notion of weak authentication and discuss the economic impacts from the attacker's point of view. SSH is a typical example which follows the leap-of-faith method in that it alerts users of key changes. However, SSH does not support transparent rollover of keys.

Applications of weak authentication are discussed in for ad-hoc wireless networks [218] and by the IPv6 community [50].

Bentley *et al.* [26] developed a security patch for sendmail which opportunistically encrypts SMTP sessions between mail transfer agents, yet does not provide end-to-end security. Stream [91] uses SMTP and POP proxies that transparently handle encryption on behalf of the MUA. It thus provides end-to-end security with a zero user interface. However, Stream does not support digital signatures nor key updates. The Enigma system of Brown and Snow [40] has a similar proxy architecture. Enigma adheres to the PGP format in contrast to Stream and our system which both encode all cryptographic information in proprietary email headers.

6.8 Conclusions

It has been widely acknowledged that security is as much a human factors problem as it is a technical one, and some authors even suggest that retrofitting usability to existing security mechanisms is no more likely to be successful than retrofitting security mechanisms to existing applications (see Section 2.3.3). In this chapter, we presented an innovative design approach towards non-intrusive secure email by adopting the common threat model. Our system prevents passive eavesdropping attacks completely and limits the impact of active attacks to cases where the adversary launches a MITM attack prior to the first email exchange. Even in the case of a successful attack, it is infeasible for the adversary to leave the communication channel undetected, because of the double signatures in key rollover messages.

As a guideline we used the prudent engineering rules for designing protection mechanisms stated in Section 6.2.2.

- In order to reduce PKI's complexity which originates from the processing of certificates, we separated key exchange (which is performed transparently) from binding keys to identities (which becomes a trade-off between usability and security that the user can make at his own discretion).

- As a fail-safe default, email is encrypted transparently and automatically whenever possible. The user must only decide how mail is handled for recipients for whom no suitable key is known.

- The need for user interaction is kept at a minimum and is non-intrusive as the user has the choice to continue her work task uninterrupted. All interaction metaphors are based on familiar concepts and mental representations.

Our main thesis, the low comparative benefit/cost ratio of certification infrastructures for private non-commercial users, and the potential scalability of pair-wise key verification, is supported by the reduction in user interactions that our approach achieves as well as by the results of a study of users' mail communication. We conclude that security and usability of email is not necessarily mutually exclusive if best practices in the design of secure and usable systems are adhered to diligently.

We regard three directions for future research as particularly promising: first, innovative human-computer interaction support for usable key verification; second, finding alternative and improved metrics to maximize users' security per interaction in a customized and adaptive fashion; third, innovative approaches to provide user interface "incentives" which promote ubiquitous "best effort" security as a base level of (mail) protection.

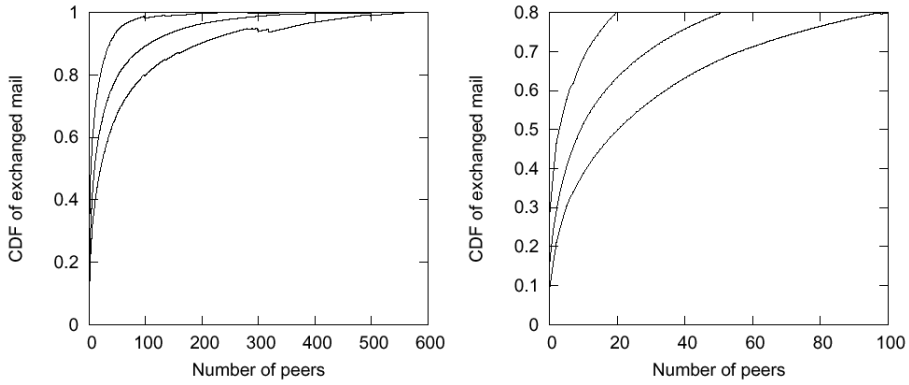

Figure 6.11: Cumulated percentage of exchanged mails (ordinate) plotted against number of peers (abscissa) with whom the mails were exchanged.

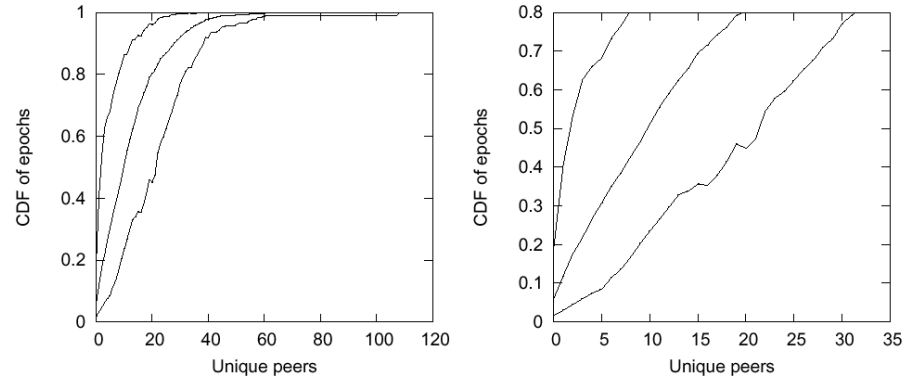

Figure 6.12: Cumulated percentage of weeks (ordinate) plotted against number of unique peers per week (abscissa).

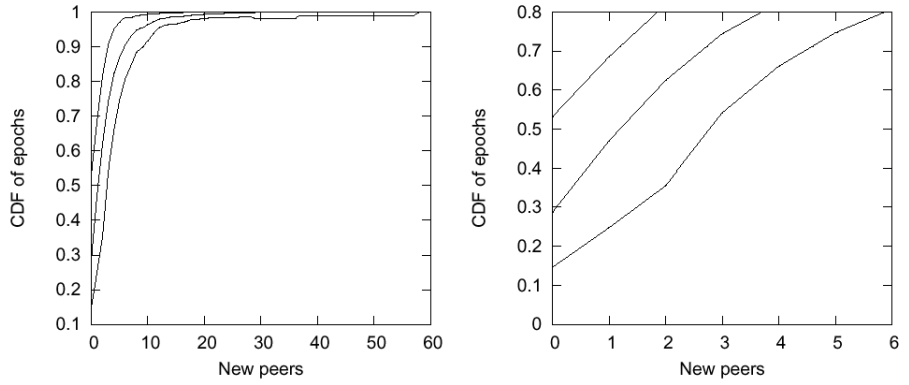

Figure 6.13: Cumulated number of weeks (ordinate) plotted against number of new peers encountered per week (abscissa).

Chapter 7

A Multipurpose Delegation Proxy for WWW Credentials

> *Put all your eggs in one basket, and then watch that basket very carefully.*
>
> – N.N.

> *Any problem in computer science can be solved with another layer of indirection.*
>
> – David J. Wheeler

The need for the delegation of private keys (PSE) has been identified as a PKI challenge in Section 3.2.5. In this chapter we present an architecture for the WWW that treats the delegation of credentials – or more precisely the ability to use them. Credentials may take on the form of a private key and a certificate or a user name/password pair. Both types are supported by our prototype in a uniform way. It supports all common authentication methods used with standard HTTP as well as with SSL. SSL connections in general and X.509 client certificate credentials in particular posed implementation challenges. The solution only requires minor changes on the client side and does not require any changes on the server side as it is completely transparent for the Web server. In addition, it provides a tool for *credential management* helping users to keep track of which services they have signed in on the Internet and what credentials they have used for.

This chapter is structured as follows: In the next section, we motivate the topic and highlight characteristic problems by means of a small scenario.

A short survey of the most common authentication methods on the WWW is given in Section 7.2, readers familiar with the subject may skip it. Four different system architectures are presented and compared according to our PKI evaluation tool of Section 3.3 in the following section. A prototype was developed as part of [235, 104], its realization is sketched in Section 7.4 (see [103] for a comprehensive description). Finally, related work is reviewed in Section 7.5.

Note that in this chapter we use the term proxy in two different ways. One the one hand, a proxy describes a person like a secretary or vacation replacement. On the other hand, a proxy is an intermediate component in electronic communications (like for instance an HTTP proxy). The meaning becomes clear from the context.

7.1 Problem Description

According to Oxford Advanced Learner's Dictionary, *credentials* are "documents showing that a person is what he claims to be, is trustworthy, etc." Similarly, the purpose of a credential in the digital world is to prove the identity of its owner towards another party, typically a server. The server itself or a dedicated back-end system subsequently assigns access rights to this identity such that the client is able to use the corresponding services.

Basically, credentials can be grouped into three classes: Authentication may be based on the *knowledge of a secret*, usually a password, a PIN, or a passphrase, which have to be provided to the system in conjunction with a user name that serves as a (public) unique identifier. Another class is authentication based on the *possession of a token*. A typical realization is a challenge-response protocol using a cryptographic key pair. In this scheme, the user proves that she possess the secret (the private key) to the server without revealing it. *Biometrics* give rise to a third class of user authentication methods, but have little importance on the WWW at the moment. Moreover, physical characteristics effectively tie a credential to its owner and prevent delegation of any kind, so we do not consider this class further.

7.1.1 Basic Scenario

The employees of ACME Inc., a medium-size IT company, periodically access a couple of web sites protected by different user authentication schemes. For instance, the company's manager, Alice, handles the corporate bank account via the Internet. The bank's web server is accessed through an SSL channel. It requires strong client authentication using a key pair and the corresponding X.509 certificate the bank had initially distributed in the form of a PKCS#12 softtoken. Alice wants her proxy Bob to manage the company's bank transactions during her summer vacation. Since she considers trust to be good, but control to be better, she also wants a way to monitor when Bob actually logged in on her behalf while she is absent.

Carl has subscribed to a commercial web portal on behalf of ACME. The services are charged on a per-request basis and require a logon with a user name and a password. The management wants to keep track of how often members of the research department use the portal and restrict access to office hours. Another requirement is a possibility to easily withdraw the authorization of engineers leaving the company. Unfortunately, ACME has only a single company-wide account ...

7.1.2 Technical and Usability Requirements

The previous scenario highlights problems with credentials that are *delegated* or *jointly used* by a group of people – these issues are independent of the concrete type of credential. By delegating a credential, its owner permits another person to temporarily impersonate her toward the target server, i.e. to adopt the virtual identity of the owner. Group usage can be regarded as a generalization of this concept. Both applications require that the capability to impersonate the credential owner can be controlled and revoked when needed. Following the naive approach of handing out the credential to the proxy or sharing it with the group members is insecure. "Revoking" a credential based on knowledge can be achieved by altering the secret, although this is awkward and is only possible if the server permits it. In general, this possibility is ruled out for authentication schemes based on the possession of a token, since it is issued by the server's operator or a third party. In both cases, there is no way to prevent proliferation of the credential, nor a guarantee that it has been deleted. There is no possibility to control when and which protected resources are accessed and by whom.

Consider again the case where a credential consisting of an X.509 certificate should be delegated. The key pair might be used for a number of purposes including secure email, file encryption, document signing etc. In this situation, it is intolerable that the proxy gets to know the private key, which is a valuable secret used over a rather long time (the certificate's lifetime may be in the order of years). We seek for a secure way to solve these problems. To sum up, a solution should meet the following requirements:

- Perform user authentication for an authorized person on behalf of the credential owner without revealing the secret to the former.
- Protect the credentials from unauthorized access.
- Support common WWW authentication mechanisms, namely Basic and Digest Authentication, NTLM HTTP Authentication, and form-based variants as well as SSL with client certificates (see Section 7.2 for details).
- Operate transparently without additional software, both on the server and the client side.
- Require no reconfiguration on the server and only minimal reconfiguration on the client side. Provide an easy-to-use interface to manage users, credentials and access rights.

The first requirement suggests to introduce a level of indirection between credentials and users that are allowed to *dispose* of some of them. To fulfil the second one, credentials have to be stored in a trusted environment and revealed (or applied in case the credential is a key pair) during the communication with the server.

7.2 User Authentication Methods on the WWW

Authentication on a TCP/IP network should take place on the transport or application layer such that a particular user (and not only her machine) is identified [70]. We outline the most common authentication methods currently used on the WWW (see [189], [86], [105], and [165] for a thorough treatment). Solutions based on Java applets or ActiveX controls are deliberately omitted as they are of minor practical importance. Apart from

NTLM HTTP Authentication (which is likely to fall in this category, too), all methods described in the following are supported by the current prototype.

7.2.1 Basic Authentication and Digest Authentication

HTTP offers two built-in authentication mechanisms, namely *Basic Authentication* and *Digest Authentication*. Both work in the following way: Each time the browser requests a protected resource, the web server returns a message with the status code `401 Unauthorized` indicating the need for user authentication and the corresponding method to be used. After having once prompted for a user name and password assigned to a certain *realm*, the browser has to include the same authentication information over and over again in each subsequent request of a protected resource. When using Basic Authentication, this information is transferred in the clear as part of the HTTP header – possibly multiple times. This makes the method extremely vulnerable to eavesdropping. In comparison, Digest Authentication is resistant to passive attacks since the password is concealed by a challenge-response protocol. The challenge consists of a server-created nonce, which has to be incorporated in the argument of a hash function (besides the user name and password) to obtain the response. Digest Authentication thus provides significantly more security concerning the authentication although it cannot withstand active attacks, nor does it provide confidentiality.

7.2.2 NTLM HTTP Authentication

NTLM HTTP Authentication is a proprietary authentication scheme working analogously to Digest Authentication, but requires an additional round of communication between client and server. Depending on the server settings, a client may answer the challenge with one or more responses using different algorithms [105]. The protocol is based on Microsoft's NTLM (NT LAN Manager), but platform-independent implementations of NTLM HTTP Authentication are available[1]. NTLM differs from the previous methods in that it does not authenticate single HTTP requests, but merely a whole session conveying session authentication information via the HTTP headers. Server and client are both required to support persistent con-

[1] e.g. by Mozilla, see `http://www.mozillazine.org/talkback.html?article=3990`.

nections either with HTTP/1.0 and the *keep-alive* feature or HTTP/1.1 (common web browsers do).

7.2.3 Authentication based on HTML Forms

Using HTML forms [186] for user authentication is another method on the application layer and certainly the most popular nowadays. From an aesthetic and usability viewpoint, this is the method of choice since the authentication dialogue can be embedded directly into the web page, thus avoiding extra windows popping up.

There are two transport methods for information that was entered into the form. While the GET method appends the data to the URL, the POST method transfers it in the body of the HTTP request. A disadvantage of the former method is that information may be unintentionally cached (e.g., in the browser's history or in log files on the web server or an HTTP proxy). Like all aforementioned methods, form-based authentication does not provide message confidentiality or authenticity by itself. Due to the stateless nature of HTTP, form-based authentication has to be used in conjunction with techniques for session management to convey authentication information in subsequent requests. HTTP cookies [149] or URL re-writing are typical such methods.

7.2.4 Public Key Methods

The methods considered so far are all knowledge-based. In order to use a stronger authentication mechanism based on the possession of a private key, SSL/TLS is the protocol of choice for the Internet. Here we do not distinguish between SSL [124, 87] and TLS [66], but simply speak of SSL instead as TLS v1.0 is closely related to SSL v3.0. SSL is already widely deployed, although mostly in a setting where only the server has a certificate, but users are authenticated by other means (those described in the previous sections). Let us assume that the user possesses a key pair for signing and a corresponding certificate issued by a certification authority (CA) that is trusted by the server.

Before the actual data transmission takes place, client and server engage in a handshake protocol to mutually authenticate and agree upon a common secret. During the protocol, the client, which holds an RSA or DSA key pair, sends an X.509 certificate and proves its identity by signing a hash code

based on the preceding messages exchanged with the server (see e.g. [219, 70] for details). As a result of the handshake, client and server have agreed upon a so-called pre-master secret. A key for a MAC scheme is derived from this value. The MAC key is used to authenticate subsequent messages.

7.3 A Comparison of System Architectures

In this section, we present four different architectures that fulfil the requirements stated in Section 7.1.2. One of these architectures is a client-side solution while the others require a server acting as a *man-in-the-middle*, which is shown in Figure 7.1. The application server, the HTTP gateway, and the HTTP proxy architecture come under this category. The illustration shows the communication between the client and server via an intermediate MITM component. The segments between client and MITM as well as between MITM and server can be secured to prevent eavesdropping and ensure data integrity (depicted by the tunnel with the padlock symbol).

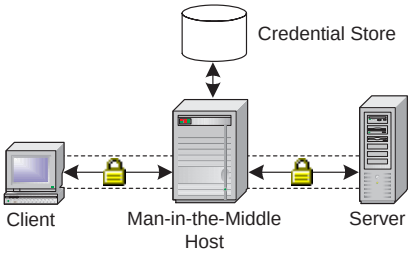

Figure 7.1: Generic man-in-the-middle architecture.

In the following, we use the terms *middleware* for the MITM component and *target* host to indicate the actual connection endpoint, i.e. the machine that serves the requested web pages. Requests from the client to the target host go through the middleware where they are augmented with authentication credentials. In particular, credentials never leave the MITM server's protected environment. In order to allow a fine-grained access control, the middleware needs to authenticate each user before permitting access to its services. The adopted authentication mechanism largely depends on the actual implementation and can range from plain-text passwords to cryptographically strong public key mechanisms.

7.3.1 Application Server

The application server concept is based on a machine that provides remote login facilities. Possible realizations range from a full-screen remote desktop (such as VNC[2]) to the forwarding of single browser windows (as it is the case with X11 forwarding). A person, who wants to make use of a certain credential, has to login to the application server first. After the session has been established, the person connects to the target server with a standard web browser running remotely on the application server. Most modern operating systems are shipped with tools for application server access, so this variant can be easily implemented. On the other hand, popular browsers such as Microsoft Internet Explorer or Mozilla provide certain functionality for credential management.

However, handling delegations properly or enforcing a consistent policy is difficult. Special precautions are necessary to ensure that a user cannot gain unauthorized access to credentials stored on the application server (and for instance export key pairs). Additionally, users need to work remotely, adding latency to the interaction and exposing them to an environment that possibly differs significantly from the one they usually work with.

7.3.2 HTTP Gateway

In the HTTP gateway approach, the middleware acts like a web server where connections to protected resources are initiated by calling a particular URL on the gateway. The gateway then retrieves the necessary credential and opens a channel to the target host. The target server's response is returned to the client to which it appears as coming from the gateway. As the URL requested by the client differs from the actual URL, the gateway needs to modify content coming from the target host in order to redirect hyperlinks contained in HTML or JavaScript documents to the gateway's host name. Otherwise, subsequent requests would directly address the target server, skip the gateway's proxy authentication, and thereby leading to an error.

A major advantage of this approach are the low deployment costs. There is no need to modify the client-side configuration and SSL can be handled as well (by equipping the gateway with a single X.509 certificate issued by a trusted CA). However, there are downsides: Binary data such as Macrome-

[2] `http://www.realvnc.com/`

dia Flash or Java bytecode makes content re-writing very difficult. Additionally, a Java applet that runs in a sandbox may only communicate with the server it was retrieved from by the client, so the applet's backward channel to the original server is cut.

7.3.3 HTTP Proxy

A natural idea is to use an HTTP proxy and enhance it with certain functionality to handle authentication information (using the terminology of RFC 2616 [80], such proxies are called "non-transparent"). HTTP proxies are a well-known concept. They are typically used as so-called caching proxies in order to reduce network traffic or as an application level gateway in conjunction with a firewall. When processing standard HTTP requests, a proxy works as forwarding agent in between the requesting client and the responding host, itself acting both as a server and client. This allows the proxy to modify requests and add authentication credentials if necessary. All HTTP-based schemes listed in the requirements (Section 7.1.2) can be handled that way.

Things get a bit more complicated with HTTPS requests. To initiate an SSL session through a proxy, there is a special command (the CONNECT method, see [157] for details). However, after a secure, i.e. encrypted and authenticated, tunnel between client and server has been established, the HTTP proxy is unable to change or eavesdrop on the data in transmission. The only way to work around this restriction is to loosen the principle of *end-to-end* security. This can be done by letting the proxy pretend toward the client that it is the target host. In practice, such a *man-in-the-middle attack* is effectively prevented through the use of a certificate that identifies the server. However, if the proxy is given a *replica* certificate that

(a) binds *the proxy's* public key to the identity of the target site and

(b) is issued by a CA which is trusted by the client,

the proxy can successfully impersonate the target host toward the client. If the proxy furthermore has access to an authentication credential (a key pair and the respective certificate), it can also impersonate the credential owner toward the target host.

7.3.4 Client-Side Architecture

The idea behind the client-side approach is to equip an HTTP user agent with additional functionality to access a credential store, e.g. via a PKCS#11 interface. After retrieving the appropriate credential from the central repository, authentication can take place as usual, even without the need to prompt for user input. This variant does not need a machine sitting in between client and target host, but only an adaptation of the web browser. For open-source browsers like Mozilla or its Firefox branch, the extension can be integrated directly in the code. It should be possible to also implement the same functionality for Internet Explorer using its plug-in and extension COM interface. However, as we have not implemented any of these extensions, it is difficult to give cost estimates. Another alternative, which is independent of a particular user agent, is to fit up the client's TCP/IP stack or socket library with support for user authentication via the credential store. For instance, on the Microsoft Windows platform, this can be accomplished by a wrapper around the Winsock or WinInet DLL. Such a wrapper has to detect connections requiring authentication, retrieve the suitable credential from the credential store and transparently perform the authentication.

The major advantage of this client-side concept is the seamless integration with the user's web browsing environment for the price of writing new code. While credential use can and should be tracked by the central credential store, the actual communication between client and server is kept private when SSL is used. Unfortunately, this comes for the price that credentials need to be revealed to the client. Therefore, a malicious client could learn the secret information.

7.3.5 Evaluation

In the following, we sum up the pros and cons of each architecture. For the comparison we use a subset of the categories of our general evaluation framework (see Section 3.3). Some of them emphasize single aspects as it is indicated below. The selection of categories includes *technical requirements* (stressing standard compliance and compatibility with existing software), *transparency of the security features*, *menus and dialogues*, *secret key handling* (comprising the handling of other types of credentials), and *installation and configuration* (especially with respect to deployment costs). A quantitative score is given in Table 7.1.

	Appl. Server	HTTP Gateway	HTTP Proxy	Client-Side
Tech. Requirements	∘	∘	+	∘/−*
Transparency	+/ − −*	∘	+	+ + /∘*
Menus&Dialogues	∘	∘	++	+
Secret Handling	−	+	+	−
Installation&Config.	+/ − −*	+ + /∘*	∘	−−

*Rating depends on the actual system environment and implementation variant, see text for details.

Table 7.1: Comparison of the four architectures.

We use the same notation for the rating as in Chapter 3. (∘ denotes an average, + and − denote slightly positive and negative results respectively. Excellent respectively insufficient grades are abbreviated by ++ and −−.) Please observe that in some cases the rating may vary according to the actual system environment and the implementation variant. Details are given in the following text.

Technical Requirements As the application server uses a standard web browser, it supports a variety of protocols and formats, with the possible exception of plug-ins or ActiveX controls that require root privileges the end user is not granted on the application server. Access to credential management facilities is generally restricted to software running directly on the application server. The limiting factor of the HTTP gateway is its need for URL rewriting as outlined in Section 7.3.2. In this regard, the HTTP proxy is superior as it only requires a user agent supporting HTTP proxies (which can be taken for granted). Besides, other applications can use the proxy's services through a standardized protocol, too. The latter statement also applies to the client-side approach with a modified TCP/IP stack or socket library (which are nevertheless no platform-independent solutions). A customized web browser gets a lower compatibility rating since it only supports a restricted set of applications.

Transparency of the Security Features Here transparency refers to the question to what extent the user gets in touch with the credential delegation infrastructure. Obviously, each of the MITM architectures requires an additional step of authentication toward the middleware. The rating for the application server approach varies significantly depending on the actual realization (see Section 7.3.1). A solution implementing a seamless integration into the user's work environment, such as X11 forwarding, receives a better rating than a classic remote-desktop solution like VNC. While HTTP gateway users can still work with their usual environment, accessing authenticated WWW resources is not seamless as it requires an additional step to point the web browser to the gateway page. In comparison, the HTTP proxy operates transparently after an initial configuration. The client-side approach is optimal if the user's favorite browser is supported, but may cause some inconvenience otherwise.

Menus and Dialogues The general usability of the application server approach heavily depends on the actual realization (e.g. if the look & feel is the same) and on network latency. While the latter concern is a minor issue for the HTTP gateway, it nevertheless confronts the user with a non-standard way of accessing protected WWW resources: Instead of using the browser's controls such as the location bar to navigate to the target address, the user has to use an HTML form provided by the gateway to navigate to the desired page. The HTTP proxy is favourable from a usability point of view as it does not require any uncommon actions. So is the client-side approach as long as the user can work with her favourite browser.

Secret Key (and Password) Handling All approaches have in common that they use a central credential repository, which is an attractive target for attackers. However, protecting a single system is still easier than caring for a lot of decentralized credential stores on the client systems, possibly having to deal with heterogeneous operating systems and user agents. A downside of the MITM architectures is the fact that payload data is available in the clear on the middleware – even if the target host is accessed via SSL. These architectures are thus vulnerable to insider attacks (e.g. by the system operator), but on the other hand offer the possibility to audit transactions. The communication between application server and client should be analogously protected against wiretapping (e.g. by SSH port forwarding). It is question-

able whether the application server environment can be sufficiently hardened to prevent attempts to read out stored credentials. Credential protection is a major weakness of the client-side approach. However, it is in fact the only solution to provide real end-to-end security for the communication between client and the target host.

Installation and Configuration On a Windows XP or an X11-based client system, deployment costs for the application server approach are low since the necessary software is already pre-installed. For all MITM architectures, the middleware's public key has to be trusted by the clients. If this key is certified by a CA that is unknown to the clients, the appropriate trust anchor has to be installed on all systems. This step can be combined with the rollout of end user certificates if such are used for authentication with the middleware. In the HTTP proxy realization, one has to have a key pair to issue replica certificates. This requires in turn a certificate with the *Basic Constraints* extension set to "CA". Commercial CAs charge a lot of money for certificates of this kind, so the HTTP proxy will probably rather rely on its own CA. A minor problem is the single change in the browser's proxy settings, which can be automated for some browsers (see Section 7.4.4). A client-side implementation comes with the highest deployment costs of all architectures as it requires software installation and configuration on all client systems.

7.4 TLS Authentication Proxy Prototype

We have implemented a prototype of the HTTP proxy variant, since we consider it superior to the other MITM architectures from a technical viewpoint and estimated its development costs lower than those of a client-side variant. After giving an overview of the proxy's modular architecture, we go into technical details in Section 7.4.2. The user interface is described in the following section, deployment issues are then discussed in Section 7.4.4.

7.4.1 Overview

An overview of the modules of the prototype, which we called *TLS Authentication Proxy*, is shown in Figure 7.2. The components can be grouped largely into two categories: A back-end and a front-end, which are shown on

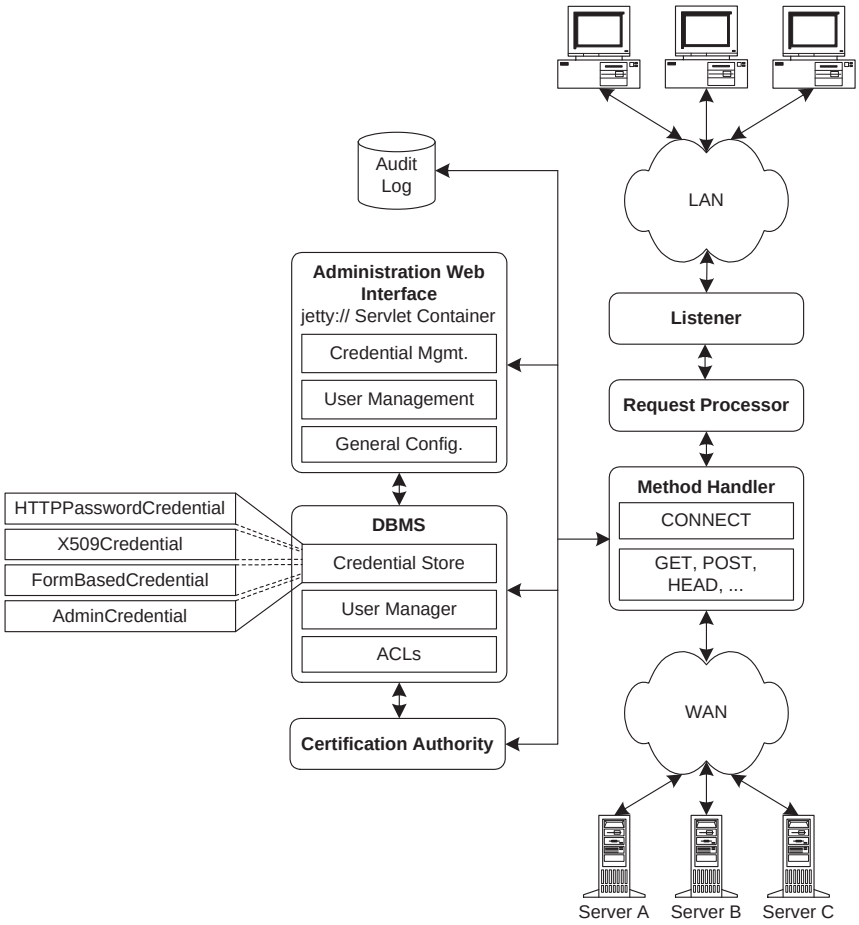

Figure 7.2: Module structure of the TLS Authentication Proxy.

the left-hand and the right-hand side of the figure, respectively. The back-end is built around the underlying database management system (DBMS) and contains modules for the management of proxy users, credentials, and access control policies. Further components are a tiny certification authority, a servlet-based web administration interface, and a logging module. The front-end comprises the networking and HTTP engines.

Our proxy is written in Java 2 SE 1.4 and uses the *FlexiProvider* JCE implementation as cryptographic service provider. ASN.1 support is provided by the *Fraunhofer Codec* package, which is relevant for parsing and issuing X.509 certificates. *PostgreSQL*[3] is used as database back-end. All

[3] `http://www.postgresql.org`

administrative tasks can be handled through a web interface, which has been implemented using the *jetty://* HTTP servlet container[4].

7.4.2 Technical Details

Upon receiving a request from the network, the proxy parses it in order to extract the target host and request method. Our implementation supports all request methods (GET, POST, CONNECT etc.) defined in RFC 2616 [80]. If the resource on the target host is known to require a credential, the proxy first authenticates the requesting client, looks up its permissions, and starts the authentication with the target host if applicable. Otherwise, TLS Authentication Proxy acts as a usual non-caching HTTP proxy. In order to identify the user, the proxy may use SSL with client authentication – which we chose to implement for the current prototype as it is the strongest mechanism – or Basic/Digest Authentication. The situation where a user requests a resource that is accessible via standard HTTP is somewhat special since it requires a redirection to an SSL page in our implementation. This case is therefore discussed later (see Figure 7.4).

Target web site accessed via SSL The necessary steps for target web sites that use SSL are shown in Figure 7.3. By means of the standard CONNECT method, a user agent indicates to the proxy that it wants to access an SSL-protected website. As a first step, TLS Authentication Proxy and the server start an SSL handshake with a premature end after the server has sent its certificate. The CA module then produces a replica of the server's just retrieved X.509 certificate. This replica is then presented to the client in a subsequent SSL handshake where the proxy impersonates the original server. During this handshake, the proxy demands certificate-based authentication from the client. Having successfully established a cryptographic channel between client and proxy, the proxy selects the credential and goes through a complete SSL handshake with the server this time. If the credential is a key pair, the proxy is already authenticated during the handshake. Otherwise, the last messages exchanged between proxy and server in Figure 7.3 look slightly different (like in Figure 7.4 for the case of Basic/Digest Authentication).

[4] `http://jetty.mortbay.org/jetty/`

Target web site accessed via standard HTTP Figure 7.4 illustrates how the proxy identifies a user who requests a protected resource via HTTP. Instead of passing through the request to `http://server/` (as a usual HTTP proxy would do), TLS Authentication Proxy redirects the web browser to the URL `https://server:9443/` forcing it to CONNECT to the SSL site. Here, the reserved port number of 9443 indicates to the proxy not to act as in the situation of Figure 7.3. When the proxy gets a subsequent request tagged with this port number, it connects to the target server using HTTP as originally intended – even though the connection between client and proxy has been secured by SSL. (The actual target port if different from the default value 80 is picked from the internal database.) A future extension might use the connection upgrade mechanism specified in RFC 2817 [143] to dynamically change an HTTP connection to HTTPS. This would allow an even more seamless user experience. However, current web browsers do not support this protocol extension yet.

Session Management To speed up successive Basic Authentication requests, the proxy maintains an authentication realm cache. Instead of having to wait for the server to return a `401 Unauthorized` status code, the proxy immediately sends the authentication headers with the request itself. This optimization is not applicable to Digest Authentication or other challenge-response schemes. In contrast, form-based authentication schemes usually do not require to re-authenticate each single request, but apply some form of session management instead. Schemes that embed session identifiers into the URL can be grouped into two categories: After successful client authentication, either an HTTP redirect (Status code `302 Found` in conjunction with a `Location` header) sends the browser to a new URL containing the session ID or the browser receives personalized web pages with the ID embedded into each hyperlink.

Our prototype comes with full support for cookie-based session management and works with the redirection-based mechanism. New sessions are established either automatically when a web site is accessed for the first time (or the previous session has expired) or manually. This comprises a so-called trigger URL, which is an arbitrary URL on the target web site the user calls to initiate authentication. URL-based sessions cannot be established automatically, but require the use of a trigger URL. TLS Authentication

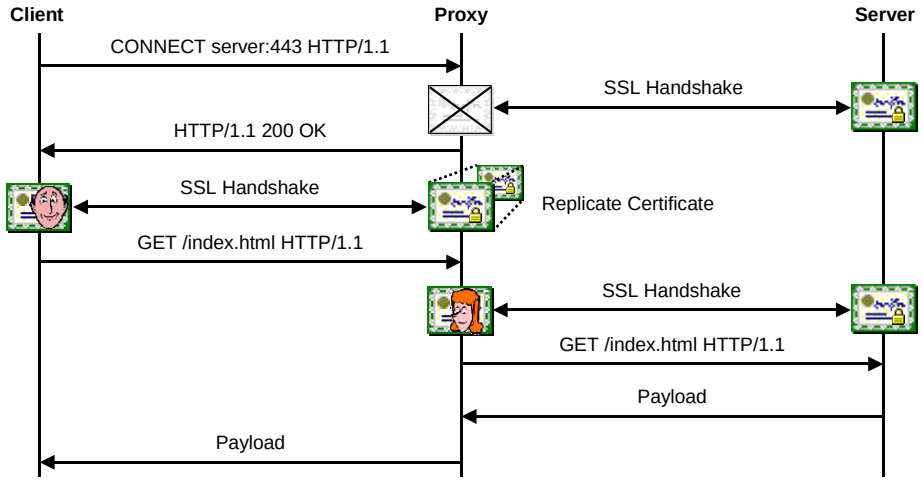

Figure 7.3: Data flow for an SSL target web site.

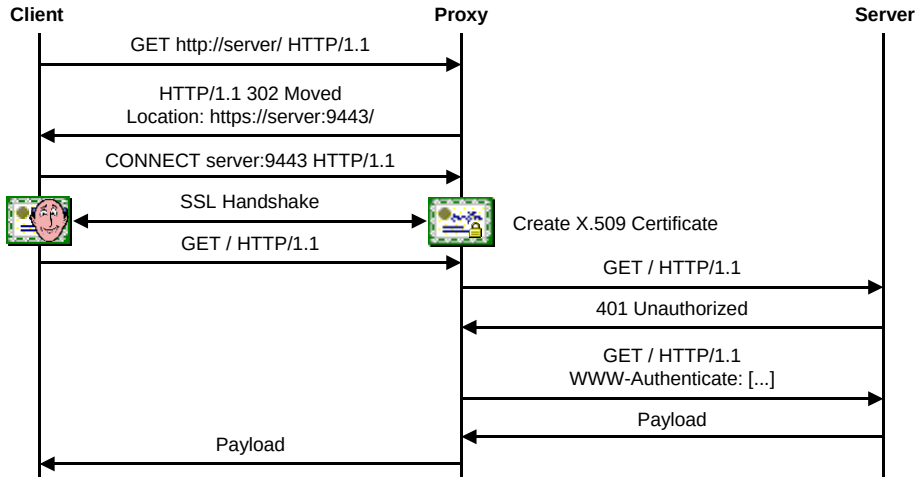

Figure 7.4: Data flow for an HTTP target web site.

Proxy keeps track of cookie-based sessions and transparently transmits the HTML authentication form data when needed. The server's login URL as well as the authentication parameters – which are usually entered into the HTML form – is stored along with the other credential information. As sessions often expire after a certain period of inactivity, form-based creden-

tials have a configurable maximum session lifetime after which the proxy automatically re-authenticates.

Logging The task of the logging module is twofold. On the one hand, it audits all access to the web interface (which is described in the following section). On the other hand, every use of a credential stored on the proxy is recorded from the back-end. Both log files adhere to the de-facto standard *NCSA Extended/Combined Log Format*[5], which is used for instance by the Apache web server. A number of tools is available to parse and evaluate such files. Among other pieces of data, the time and date of access, the name of the authenticated user, and the requested resource and credential is kept. In the current version of the prototype, this information is not yet available to end users, but only to an administrator.

7.4.3 User Interface

TLS Authentication Proxy offers a web-based user interface, which is accessible via the reserved URL `https://proxyadmin/`. Logins to this site always require a user certificate issued by the proxy's CA. The authentication process is similar to the way how other protected resources are accessed. End users can browse this site to learn which credentials they are allowed to use and delegate. Assume that Alice delegates a credential to Bob. Along with this information, Alice may define a time frame in which Bob has access to the service assigned to this particular credential. This is the only constraint that is currently available, but others are conceivable, e.g. a maximum number of total/daily logins or a restriction to a subset of web pages. Due to the modular implementation, custom constraints can be added easily. If necessary, the right to use a credential can be revoked on a per-user basis. Credential usage rights are modelled by a list of capabilities each supplemented with a constraint. Assigned to a credential is the person that has the right to delegate it.

Administration Interface Whether a person may also access the proxy's administration functionality is determined by a predefined *AdminCredential*, which can be added to a capabilities list like any other credential. Only persons who have been granted the right to use this special credential may access

[5] `http://hoohoo.ncsa.uiuc.edu/docs/setup/httpd/LogOptions.html`

the administration GUI. Apart from general network and proxy settings (for instance the address ranges that are allowed to connect), the administration interface provides functionality for user and credential management. Creating a new user goes along with the instant generation of a key pair and the issuance of a certificate for that person. Both objects are contained in a PKCS#12 file that the administrator can download together with the corresponding transport PIN. In the current implementation, certificates are renewed manually via the administration interface. Instead of implementing certificate revocation lists, we chose to give the administrator the ability to disable users temporarily or permanently, which is more flexible.[6] Figure 7.5 shows an example of an X.509 credential for the *SAP Service Marketplace*[7] imported into the proxy. The setup of form-based authentication is a bit more complicated since the parameter names and values still have to be extracted manually from the HTML source code. However, this could be automated with some effort.

7.4.4 Deployment

There are two deployment problems with the HTTP proxy approach. One is the distribution of keys and certificates, the other is web browser configuration. Key pairs and certificates are delivered within a PKCS#12 container file, which also contains the certificate of the proxy's CA. When importing a personal key pair using Internet Explorer and the Windows CryptoAPI, the CA certificate is automatically installed, too. In contrast, Mozilla requires an additional step of explicitly introducing a new trust anchor to the system.

In order to further minimize deployment costs, our prototype supports the *Web Proxy Auto-Discovery* (WPAD) Protocol [95], which can be used to automatically obtain a *Proxy Auto-Configuration* (PAC, see [172]) file from the network. The PAC file contains JavaScript code, which dynamically tells the browser the URLs to use a proxy server for and what the address of that server is. WPAD is directly supported by Internet Explorer, a patch for Mozilla is available. JavaScript is sometimes seen as a security risk and therefore deactivated. As an alternative, either the proxy server's address or the URL of the PAC file can be configured manually in the browser.

[6] For security reasons, certificates issued by the proxy's CA carry a critical extension restricting their usage to the purpose of client authentication.

[7] This portal is located at the URL `https://www010.sap-ag.de/`.

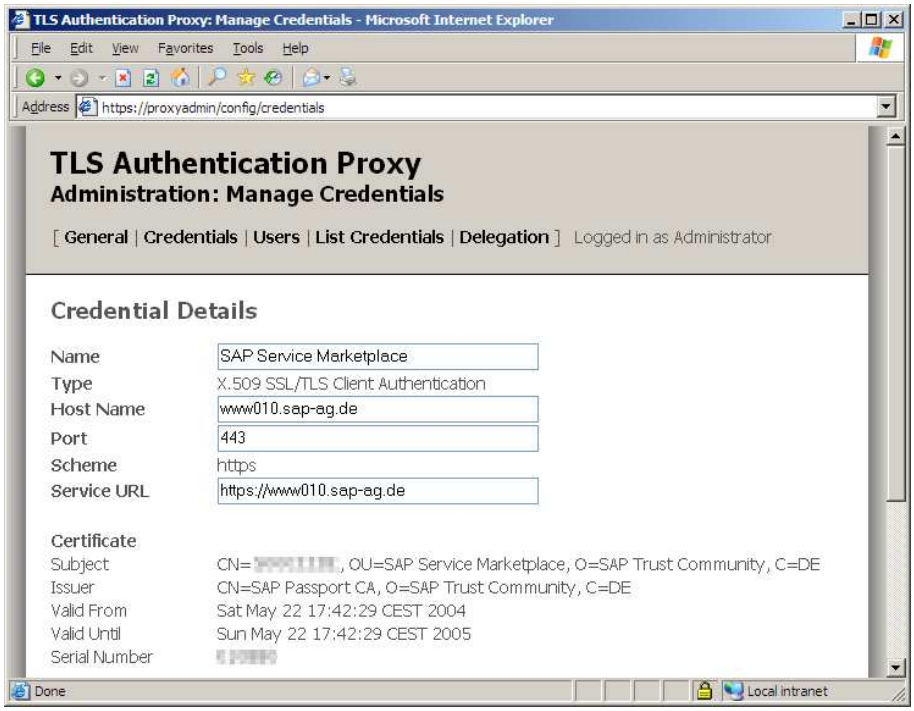

Figure 7.5: Details for the SAP Service Marketplace credential.

To counter privacy concerns, the PAC file can be adjusted to only route traffic through the proxy if the target host requires a credential. Due to the way PAC works, the configuration files can be obtained anywhere in the local network and allow an adversary to learn which credentials are kept in the proxy's repository. As a countermeasure, the proxy is able to apply a cryptographic hash algorithm as a one-way function in order to obscure the host names. The prototype uses Paul Johnston's JavaScript implementation of MD5[8]. At runtime, the PAC file's code is executed to compute the hash of the current server name. If and only if the result matches one of the stored values, the proxy is used.

7.5 Related Work

The problem of credential management typically leads to identity management or *Single sign-on* (SSO) technologies. SSO provides a means to reduce

[8] `http://pajhome.org.uk/crypt/md5/`

the number of authentications a user has to go through. After an initial authentication with an authentication service provider (ASP), protected resources can be accessed without further authentication. Surveys on SSO architectures can be found in [182, 61, 21]. SSO solutions can be categorized depending on whether the target host is aware of the ASP's involvement or not. In the notion of [182], *true SSO* services like Microsoft's Passport[9] require an established trust relationship between the ASP and the target web server, as well as the implementation of the appropriate protocol. Opposed to this are *pseudo-SSO* services.

SSO infrastructures form an ideal basis to implement a delegation mechanism. For instance, Kerberos supports this feature, but of course Kerberos was originally not designed as an authentication mechanism for the Web. However, there is in fact an Apache module[10] for the so-called *HTTP Negotiate* authentication method. Microsoft Internet Explorer and the Mozilla browsers support this method, too.[11] Kornievskaia *et al.* [148] propose a system to access "kerberized" services through a web browser in order to leverage the credential delegation functionality natively provided by Kerberos for WWW resources.

SPKI/SDSI certificates [72, 73] are designed to support delegation originally. *Greenpass* is an application that uses SPKI/SDSI certificates in an X.509 certificate environment to express the delegation of wireless network access rights. A description is found in the paper of Goffee *et al.* [106]. Greenpass is compatible with standard clients, but nevertheless required a re-implementation of the authentication protocol on the server side. The current version only supports X.509 certificates as credentials, but no password-based schemes.

Credential delegation is a natural requirement in Grid Computing as programmes should be able to run autonomously on a user's behalf with a subset of her rights. The *Grid Security Infrastructure* (GSI) uses X.509 credentials in combination with the SSL protocol to mutually authenticate Grid users and resources [46]. An entity can delegate rights to a third party with the help of X.509 proxy certificates [242] that have their own key pair as well as a particular X.509 extension and a rather short validity period.

[9] `http://www.microsoft.com/net/services/passport/`

[10] `http://modauthkerb.sourceforge.net/`

[11] Kerberos authentication is already supported natively by Mozilla and Firefox on UNIX platforms, a Windows plug-in is available at `http://negotiateauth.mozdev.org/`.

MyProxy[12] is a credential repository designed to work with a Grid portal that provides services for delegation management [21]. Weaker mechanisms like Basic or Digest Authentication are not supported. An advantage of this approach is the fact that users do not have to reveal their long-term credentials for delegation. However, the fact that this framework requires (heavy-weight) server-side support is an important downside that excludes this framework for our scenario. The same restrictions apply to the attribute certificate framework defined in the X.509 standard [52]. This approach requires a sophisticated Privilege Management Infrastructure (PMI) and limits the capability of delegation to so-called attribute authorities (AA) which are typically not end entities. Plus, the attribute certificate profile [77] explicitly discourages from the use of attribute certificate for delegation due to the high complexity of the verification process.

Impostor[13] follows a similar approach as TLS Authentication Proxy in that it performs a MITM attack on SSL. Working as an SSO proxy, it also provides content filtering, but does not issue its own replica certificates (see below).

7.6 Conclusions

In this chapter we have presented different architectures to implement a revocable delegation of WWW credentials for today's most common authentication methods on the Internet. Our work was especially motivated by the difficulty of delegating X.509 credentials, which we consider an important requirement in the near future. We have implemented an HTTP proxy with enhanced functionality, the TLS Authentication Proxy. It allows an efficient delegation and secure group usage of credentials. A major benefit of our approach is that only minimal configuration changes and no changes at all on the server side are required. TLS Authentication Proxy is a zero footprint solution as no additional software is required neither on the client nor the server side.

Credentials are stored in a central, well-protected database rather than spread over many heterogeneous client systems. Delegation of credentials can happen in a secure and revocable manner as client systems never gain

[12] `http://grid.ncsa.uiuc.edu/myproxy/`
[13] `http://impostor.sourceforge.net/`

knowledge of them. Auditing facilities log who actually delegates or uses a credential, when and for what purpose. In our opinion, the light-weight TLS Authentication Proxy represents a good compromise between usability and security.

Additionally, our prototype is a pseudo-SSO tool and facilitates the deployment of new credentials via a centralized roll-out. TLS Authentication Proxy also supports roaming scenarios where users move around among different machines. Surely, in this use case the current X.509 user authentication is not optimal, but the implementation of a one-time password mechanism is conceivable.

Chapter 8

Summary and Outlook

Wo aber Gefahr ist,
wächst das Rettende auch.

– Friedrich Hölderlin

In this thesis we proceeded along the lines of our four-stage research agenda stated in the introduction.

Stage 1: Description of Usability Problems At first, we showed numerous examples of usability problems with secure applications in Chapter 2. We then provided a *fine-grained typification* and distinction between user-caused violations and application-caused vulnerabilities, shaded by the degree of intent and impossibility respectively. The main contribution of this chapter is a multi-layer model of methods to promote usable security. We also reviewed previous work and classified it according to our scheme.

Stage 2: Particularities of PKI In Chapter 3 we delved into the details of cryptography and PKI. This lead us to the identification of *five major technical challenges* with PKI, namely certificate validation, the management of trust anchors, certificate policies, information and service access, and PSE management. For some of the challenges we proposed heuristics that could right now alleviate users' problems without changing the technology itself. Our generic *PKI evaluation tool* was also introduced. We showed its flexibility and practicability by specifying it for and applying it to a large variety of PKI use cases. We used heuristic evaluation and a cognitive walkthrough

as informal and a laboratory user test as an empirical usability evaluation method based on our framework.

Stage 3: Analysis of Generic Strategies Our *multi-layer model* represents an integral and systematic view on usable security. Having the various layers in mind helps to find the right, i.e. effective and practical, approach when designing for better usability in a security context. Solutions on the lower layers are in general more likely to allow farther-reaching improvements than solutions on the upper layers (take our concept of opportunistic email security as an example). However, this may come for the price of a radical change of technology and lead to non-standard and incompatible mechanisms, which will not be adopted in the short term. For the meantime, retrofitting usability to secure applications may be an option as we showed in Chapter 7.

Stage 4: Sample Solutions Among the other solutions is the proposal of a new concept for *security awareness* campaigns. In order to initiate a change in users' behaviour we resort to a concept called pentest@work to let users experience potential dangers first hand. We expect that such campaigns, which put concrete experience at the first place and combine different learning methods, achieve a higher and longer-lasting effect than previous approaches. This conjecture is backed by findings from learning psychology, a method to validate its soundness is also outlined.

We believe that, for the medium term of PKI development, especially rethinking the threat model merits careful attention. With PKI outsourcing (Chapter 5) and opportunistic email security (Chapter 6), we proposed two solutions in this direction. We recognized the potential of *opportunistic security for email*, where it could bring a great improvement and dramatically raise the ratio of encrypted and signed email with a transparent and non-intrusive mechanism. The current discussion in the European Union about the possible data retention of email traffic information may spur users to care for their online privacy. Opportunistic security separates key exchange from binding keys to identities. In case where the communicants require such a binding, they may engage in bilateral key fingerprint verification. A quantitative measurement in an experimental analysis of users' mailboxes suggests that this is indeed feasible. On the user interface level, we proposed

new interaction mechanisms and metaphors, which were successfully tested in a paper-based user study.

We argue that the concept of opportunistic security could also be extended to SSL as this comes close to the way people cope today with SSL server certificates that cannot be validated. The integration of the corresponding opportunistic mechanism into one of the current open-source web browsers combined with a field test would be an interesting direction of further research. We also showed how to cope with another shortcoming of SSL. Web portals that require user authentication via a certificate or other means do usually not support *secure credential delegation*. Practical needs, however, may dictate that users give another person the right to impersonate them temporarily, but without handing out the secret. This is impossible without an appropriate server-side support. Our solution, TLS Authentication Proxy, shows how to seamlessly retrofit the missing functionality at the price of loosen the principle of end-to-end security.

Gateway-based secure email is another application where end-to-end security is abandoned for the sake of better usability. We proposed a new method to secure, with the help of threshold cryptography, the central server, which otherwise becomes a single point of attack. The possibility to enforce a four-eyes principle also turned out to be useful for the case of *delegating access to encrypted mail*. Our main application for threshold cryptography was the *PKI outsourcing scenario*. Using distributed signatures we fixed a weakness in the common enrolment protocol by giving back full control to the customers. We proposed several flexible combinations of signature algorithms, certificate hierarchies, and integration variants. In order to not require the customer to rely on a third party, we also needed a distributed key generation algorithm. As a theoretical "by-product" we devised a new algorithm for the two-party case of RSA, which is asymptotically faster than previous schemes and in fact turned out to be practical.

Concluding Remarks It is unlikely that the adoption of usable PKI-enabled applications will happen quickly along a wide front without significant technological improvements. On the contrary many small steps are needed towards that goal. As first measures application designers should aim at ruling out sloppiness and apply heuristics to simplify otherwise difficult tasks of PKI-enabled software. Especially they must avoid asking questions

the user cannot answer. This is neither useful from a security perspective, nor does it strengthen users' confidence in the mechanisms themselves.

In the medium or long term, new PKI security paradigms should be looked for and standardized. We mentioned threshold cryptography and opportunistic security as two promising examples. The secure mode of operation should become the default mode and threat awareness should be raised. This would probably lead to a stronger usage of PKI-enabled applications and, due to the growing market, software quality would improve. Eventually this would result in an overall reduction of security risks.

Bibliography

[1] *ISO 15408: Common Criteria for Information Technology Security Evaluation (CC) Version 2.0*. 1998.

[2] Leichtsinn bei Passwörtern ermöglicht Datenklau. *CIO Online*, (30), 2005.

[3] Notierte Passwörter sind oft sicherer. *Computerzeitung 2005-07-25*, 30, 2005.

[4] A. Adams and M. A. Sasse. Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures. *Communications of the ACM*, 42(12):40–46, 12 1999.

[5] Anne Adams, Martina Angela Sasse, and Peter Lunt. Making passwords secure and usable. In *Proceedings of the HCI'97 Conference on People and Computers*, pages 1–19, 1997.

[6] C. Adams, P. Sylvester, M. Zolotarev, and R. Zuccherato. Data validation and certification server protocols. RFC 3029, February 2001.

[7] Carlisle Adams. Keynote Speech at the 2nd EuroPKI Workshop, June 2005.

[8] Carlisle Adams and Steve Lloyd. *Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations*. Number 2. New Riders Publishing, 2000.

[9] J. Anderson. Information security in a multi-user computer environment. *Advances in Computers*, pages 1–35, 1973.

[10] R. Anderson and T. Lomas. Fortifying key negotiation schemes with poorly chosen passwords. *Electronics Letters*, 30(13):1040–1041, 1994.

[11] Ross Anderson. Why information security is hard – an economic perspective. 2001.

[12] Anti-Phishing Working Group. Phishing activity trends report. `http://www.antiphishing.org`. 11/2004.

[13] H. Appel, I. Biehl, A. Fuhrmann, M. Ruppert, T. Takagi, A. Takura, and C. Valentin. Ein sicherer, robuster Zeitstempeldienst auf der Basis verteilter RSA-Signaturen. In P. Horster, editor, *DuD Fachbeiträge*, pages 243–250, Braunschweig, 2000. vieweg.

[14] Jari Arkko and Pekka Nikander. How to authenticate unknown principals without trusted parties. In *Proc. Security Protocols Workshop*, Cambridge, UK, 2002.

[15] Harald Baier, Judith Klink, and Tobias Straub. *Leitfaden zum Einsatz digitaler Signaturen.* hessen-media, 2003.

[16] Harald Baier and Tobias Straub. Awareness by Doing – ein neues Konzept zur Sensibilisierung von IT-Anwendern. In *Proc. BSI-Kongress*, pages 313–328, Bonn, 2005.

[17] D. Balfanz, G. Durfee, R. Grinter, and D. K. Smetters. In search of usable security: Five lessons from the field. *IEEE Security & Privacy*, 2(5):19–24, September/October 2004.

[18] Dirk Balfanz. Usable Access Control for the World Wide Web. pages 406–415, 2003.

[19] Helmut Balzert. *Lehrbuch der Software-Technik.* Spektrum, 1998.

[20] Petra Barzin and Stefan Kelm. Erstellung eines Gutachtens zur Vorbereitung und Bewertung der Ausschreibung einer PKI V2. Technical report, Fraunhofer-Gesellschaft, 2005.

[21] Jim Basney, William Yurcik, Rafael Bonilla, and Adam Slagell. Credential Wallets: A Classification of Credential Repositories Highlighting MyProxy. In *Proc. 31st Research Conference on Communication, Information and Internet Policy*, September 2003.

[22] D.E. Bell and L.J. LaPadula. Secure computer systems: Mathematical foundations and model. Technical report, The MITRE Corp., Bedford MA, 5 1973.

[23] Stephen Bell. Invalid banking cert spooks only one user in 300. `http://www.pcworld.idg.com.au/index.php/id;1998944536;fp;2;fpid;1`, May 2005.

[24] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault tolerant distributed computation. In *Proc. 20th ACM Symposium on Theory of Computing*, pages 1–10, Chicago, USA, 1988.

[25] J. Benaloh. Dense probabilistic encryption. In *Proc. Selected Areas of Cryptography (SAC'94)*, pages 120–128, Kingston, Canada, 1994.

[26] D. Bentley, G. Rose, and T. Whalen. ssmail: Opportunistic encryption in sendmail. *Proceedings of the Thirteenth Systems Administration Conference (LISA XIII) (USENIX Association: Berkeley, CA)*, page 1, 1999.

[27] Alfred Beutelspacher, Jörg Schwenk, and Klaus-Dieter Wolfenstetter. *Moderne Verfahren der Kryptografie*. vieweg, 2004.

[28] I. Biehl and T. Takagi. A New Distributed Primality Test for Shared RSA Keys Using Quadratic Fields. In *Proc. 7th Australasian Conference on Information Security and Privacy (ACISP'02)*, pages 1–16, Melbourne, Australia, 2002.

[29] S. Blackburn, S. Blake-Wilson, M. Burmester, and S. Galbraith. Shared Generation of Shared RSA Keys. Technical Report CORR 98-19, University of Waterloo, Canada, 1998.

[30] D. Boneh. Twenty Years of Attacks on the RSA Cryptosystem. *Notices of the American Mathematical Society*, 46(2):203–213, 1999.

[31] D. Boneh, X. Ding, G. Tsudik, and M. Wong. A method for fast revocation of public key certificates and security capabilities. In *Proc. 10th Usenix Security Symposium*, 2001.

[32] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Proc. Advances in Cryptology: CRYPTO'97*, pages 425–439, Santa Barbara, USA, 1997.

[33] D. Boneh and M. Franklin. Identity Based Encryption From the Weil Pairing. In *Proc. Advances in Cryptology – Cryto'01*, number 2139 in LNCS, pages 213–229. Springer, 2001.

[34] D. Boneh and J. Horwitz. Generating a Product of Three Primes with an Unknown Factorization. In *Proc. Algorithmic Number Theory, Third International Symposium, ANTS-III*, pages 237–251, Portland, USA, 1998.

[35] D. Boneh and H. Shacham. Fast variants of RSA. *CryptoBytes*, 5(1):1–9, 2002.

[36] C. Boyd. *Digital multisignatures*. Clarendon Press, Oxford, 1989.

[37] R. Brandner and U. Pordesch. Long-Term Conservation of Provability of Electronically Signed Documents. In *Proc. Information Security Solutions Europe*, Paris, 2002.

[38] Lars Brückner, Jan Steffan, Wesley Terpstra, and Uwe Wilhelm. Active data protection with data journals. In Rüdiger Grimm, Hubert B. Keller, and Kai Rannenberg, editors, *Proc. INFORMATIK 2003*, pages 269–280, 2003.

[39] Sacha Brostoff and M. Angela Sasse. Safe and sound: a safety-critical approach to security. In *Proc. NSPW*, 2001.

[40] I. Brown and C.A. Snow. A proxy approach to e-mail security. *Software – Practice and Experience*, 29(12):1049–1060, 1999.

[41] Johannes Buchmann. *Cryptographic Protocols*. (Lecture Notes), TU Darmstadt, 2002.

[42] Johannes Buchmann. *Introduction to Cryptography*. Springer, 2004.

[43] Johannes Buchmann. *Public-Key Infrastrukturen*. (Lecture Notes), TU Darmstadt, 2005.

[44] Johannes Buchmann, Harald Baier, and Tobias Straub. *Absicherung von Anwendungen mit der Unterstützung von Public-Key-Infrastrukturen*. Usability Study on behalf of Microsoft Deutschland GmbH, 2003.

[45] Bundeszentrale für gesundheitliche Aufklärung. Aids im öffentlichen Bewusstsein der Bundesrepublik Deutschland 2002. `http://www.bzga.de/studien/`, 2002.

[46] R. Butler, V. Welch, D. Engert, I. Foster, S. Tuecke, J. Volmer, and C. Kesselman. A national-scale authentication infrastructure. *IEEE Computer*, 33(12):60–66, 2000.

[47] Gaius Iulius Caesar. *De bello Gallico*. Reclam, 1980.

[48] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP Message Format. RFC 2440, 1998.

[49] Ran Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, The Weizmann Institute of Science, 1995.

[50] Claude Castelluccia, Gabriel Montenegro, Julien Laganier, and Christoph Neumann. Hindering Eavesdropping via IPv6 Opportunistic Encryption. In *ESORICS*, pages 309–321, 2004.

[51] CCITT. X.500 Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services. Standard.

[52] CCITT. Recommendation X.509: The Directory – Authentication Framework. Standard, 1988.

[53] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Comm. of the ACM*, 24:84–88, 1981.

[54] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proc. 20th STOC*, pages 11–19, 1988.

[55] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 3647, 2003.

[56] C. Cocks. Split knowledge generation of RSA parameters. In *Cryptography and Coding: Proc. 6th IMA Conference*, pages 89–95, Circencester, UK, 1997.

[57] C. Cocks. Split Generation of RSA Parameters with Multiple Participants. Technical report, 1998. URL: `http://www.cesg.gov.uk`.

[58] D.H. Crocker. Standard for ARPA Internet Text Messages. RFC 822, 1982.

[59] E. Crowley. Experiential learning and security lab design. In *Proc. SIGITE*, pages 169–176, Salt Lake City, 2004.

[60] Don Davis. Compliance defects in public key cryptography. In *Proc. 6th USENIX Security Symposium*, pages 171–178, 1996.

[61] Jan De Clercq. Kerberized Credential Translation: A Solution to Web Access Control. In *Proc. InfraSec 2002, LNCS 2437*, pages 40–58. Springer-Verlag, 2002.

[62] DFN-CERT. Nicht authentische Zertifikate der "Microsoft Corporation". `http://cert.uni-stuttgart.de/archive/win-sec-ssc/2001/03/msg00042.html`, 2001.

[63] DFN-CERT. Schwachstelle in der Zertifikatsprüfung durch MS CryptoAPI – MS02-050. `http://cert.uni-stuttgart.de/archive/win-sec-ssc/2002/09/msg00008.html`, 2002.

[64] Rachna Dhamija, Hao Chen, and Manuel Figallo. Iterative Design of a Cryptographic Key Management User Interface. `http://www.sims.berkeley.edu/courses/is213/s99/Projects/P4/`, 1999.

[65] Rachna Dhamija and Adrian Perrig. Déjà vu: A user study. using images for authentication. In *Proceedings of the 9th USENIX Security Symposium*, 2000.

[66] T. Dierks and C. Allen. The TLS Protocol – Version 1.0. RFC 2246, January 1999.

[67] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[68] Roger Dingledine. Talk at DIMACS SOUPS workshop, 2004.

[69] Paul Dourish and David Redmiles. An Approach to Usable Security Based on Event Monitoring and Visualization. In *Proc. New Security Paradigms Workshop*, pages 75–81, Virginia Beach, 2002.

[70] Claudia Eckert. *IT-Sicherheit*. Oldenburg, 2003.

[71] W. K. Edwards, M. W. Newman, J. Z. Sedivy, T. F. Smith, D. Balfanz, D. K. Smetters, H. C. Wong, and S. Izadi. Using speakeasy for ad hoc peer-to-peer collaboration. In *Proceedings of ACM 2002 Conference on Computer Supported Cooperative Work (CSCW 2002)*, New Orleans, LA, 2002.

[72] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B.M. Thomas, and T. Ylonen. Simple public key certificate. IETF Internet Draft, July 1999. URL: `draft-ietf-spki-cert-structure-06.txt`.

[73] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B.M. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, September 1999.

[74] Carl Ellison. Establishing identity without certification authorities. In *Proc. Sixth USENIX Security Symposium*, July 1996.

[75] Ernst & Young. Global Information Security Survey. `http://www.ey.com/global/download.nsf/International/TSRS_-_Global_Information_Security_Survey_2003/`, 2003.

[76] Wen Fang and Stephan Wappler. LDAP Proxy: Einfacher Zugriff auf Verzeichnisdienste. In *Proc. D-A-CH Security*, 2004.

[77] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC 3281, April 2002.

[78] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In *Proc. 28th Annual Symposium on the Foundations of Computer Science*, 1987.

[79] D.C. Feldmeier and P.R. Karn. Unix password security – ten years later. In *Advances in Cryptology – CRYPTO '89 (LNCS 435)*, pages 44–63, 1990.

[80] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.

[81] Ivan Flechais, M. Angela Sasse, and Stephen M. V. Hailes. Bringing security home: A process for developing secure and usable systems. In *Proc. NSPW*, 2003.

[82] FlexSecure GmbH. FlexiTrust 3.0 ReCrypt Tool for MS Outlook. `http://www.flexsecure.de`, 2005.

[83] Scott Flinn. SSL & the average person. Post on the HCISec mailing list, July 2005.

[84] D. Fox. Security Awareness oder: Die Wiederentdeckung des Menschen in der IT-Sicherheit. *Datenschutz und Datensicherheit*, 27:676–680, 2003.

[85] Dirk Fox and Sven Kaun. Security Awareness-Kampagnen. In *Proc. BSI-Kongress*, pages 329–338, 2005.

[86] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617, June 1999.

[87] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL Protocol – Version 3.0. Internet draft, Netscape Communications Corp., November 1996. URL: `http://wp.netscape.com/eng/ssl3/draft302.txt`.

[88] Batya Friedman, David Hurley, Daniel C. Howe, Edward Felten, and Helen Nissenbaum. Users' Conceptions of Web Security: A Comparative Study. In *Proc. CHI 2002*, pages 746–747, 2002.

[89] Arnulph Fuhrmann. Verteilte Effiziente Schlüsselgenerierung in Java. Studienarbeit 7/2000.

[90] Simson L. Garfinkel. Email-Based Identification and Authentication: An Alternative to PKI? *IEEE Security & Privacy*, pages 20–26, 2003.

[91] Simson L. Garfinkel. Enabling email confidentiality through the use of opportunistic encryption. In *Proc. National Conference on Digital Government Research*, 2003.

[92] Simson L. Garfinkel. How to Make Secure Email Easier To Use. In *Proc. CHI*, 2005.

[93] Simson L. Garfinkel and Robert C. Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *Proc. of the 2005 symposium on usable privacy and security*, 2005.

[94] Simson L. Garfinkel, Jeffrey I. Schiller, Erik Nordlander, David Margrave, and Robert C. Miller. Views, reactions and impact of digitally-signed mail in e-commerce. In *Proc. Financial Cryptography and Data Security*, 2005.

[95] Paul Gauthier, Josh Cohen, Martin Dunsmuir, and Charles Perkins. Web Proxy Auto-Discovery Protocol. Internet draft, December 1999. URL: `http://www.web-cache.com/Writings/Internet-Drafts/draft-ietf-wrec-wpad-01.txt`.

[96] R. Gennaro, S. Jarecki, H. Krawczy, and T. Rabin. Robust Threshold DSS Signatures. In *Proc. Advances in Cryptology – Eurocrypt'96*, 1996.

[97] D. Gerd tom Markotten. User-Centered Security Engineering. In *Proc. fourth EurOpen/USENIX Conference – NordU2002*, February 2002.

[98] D. Gerd tom Markotten. *Benutzbare Sicherheit in informationstechnischen Systemen.* PhD thesis, Universität Freiburg, 2003.

[99] Daniela Gerd tom Markotten, Uwe Jendricke, and Günter Müller. Benutzbare Sicherheit – Der Identitätsmanager als universelles Sicherheitswerkzeug. In Günter Müller and Martin Reichenbach, editors, *Sicherheitskonzepte für das Internet.* Springer, 2001.

[100] Daniela Gerd tom Markotten and J. Kaiser. Usable security – challenges and model for e-commerce systems. *Wirtschaftsinformatik*, 6:531–538, 2000.

[101] R. W. Gerling and S. Kelm. E-Mail-Verschlüsselungsproxies in der Praxis. In *Proc. 11th DFN-CERT Workshop Sicherheit in vernetzten Systemen*, Hamburg, Germany, 2004.

[102] N. Gilboa. Two party RSA key generation. In *Proc. Advances in Cryptology: CRYPTO'99*, pages 116–129, Santa Barbara, USA, 1999.

[103] Thilo-Alexander Ginkel. Entwurf und Implementierung eines Authentifikations-Proxys für das World Wide Web. Diploma thesis, TU Darmstadt, July 2004.

[104] Thilo-Alexander Ginkel and Tobias Straub. Secure Delegation of WWW Credentials: A Practical Approach. In *18th DFN-Arbeitstagung*, 2004.

[105] Eric Glass. The NTLM Authentication Protocol. 2003. URL: `http://davenport.sourceforge.net/ntlm.html`.

[106] N. C. Goffee, S. H. Kim, S. Smith, P. Taylor, M. Zhao, and J. Marchesini. Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In *Proceedings 3rd Annual PKI R&D Workshop*, pages 16–30, 2004.

[107] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, 2001.

[108] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, 2004.

[109] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[110] Nathaniel S. Good and Aaron Krekelberg. Usability and privacy: a study of KaZaa P2P file-sharing. pages 137–144, 2003.

[111] Rebecca E. Grinter and D.K. Smetters. Three Challenges for Embedding Security into Applications. In *CHI Workshop on Human-Computer Interaction and Security Systems*, Fort Lauderdale, FL, April 2003.

[112] Anti-Phising Working Group. Phishing activity report may 2005, May 2005.

[113] META Group. *IT-Security im Jahr 2003 – Status, Trends und Strategien*. 2003.

[114] Peter Gutmann. X.509 Style Guide. `http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt`, October 2000.

[115] Peter Gutmann. PKI Technology Survey and Blueprint. `http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitech.pdf`, 2003.

[116] Peter Gutmann. Plug-and-play PKI: A PKI your mother can use. In *Proc. 12th USENIX Security Symposium*, Washington, DC, USA, August 2003.

[117] Peter Gutmann. Why Isn't the Internet Secure Yet, Dammit? In *Proc. AusCERT Asia Pacific Information Technology Security Conference*, 2004.

[118] Peter Gutmann. Another example of people ignoring invalid(-seeming) certificates. Post on the HCISec mailing list, July 2005.

[119] Peter Gutmann. Inadvertent case study in SSL server cert effectiveness. Post on the HCISec mailing list, May 2005.

[120] J. Hackmann. Unternehmen lagern Sicherheit ungern aus. *Computerwoche*, 2003. `http://www.cowo.de/index.cfm?pageid=256\&artid=52316`.

[121] Heise Online. Erfolgreicher Angriff auf iTAN-Verfahren. `http://www.heise.de/newsticker/meldung/66046`, November 2005.

[122] R. Henning. Security service level agreements: Quantifiable security for the enterprise? In *Proc. New Security Paradigm Workshop*, pages 54–60, Ontario, Canada, 2000. ACM.

[123] Amir Herzberg and Ahmad Gbara. Trustbar: Protecting (even naïve) web users from spoofing and phishing attacks. Cryptology ePrint Archive, Report 2004/155, 2004. `http://eprint.iacr.org/`.

[124] K. Hickman. SSL 2.0 Protocol Specification. Technical report, Netscape Communications Corp., November 1994. URL: `http://wp.netscape.com/eng/security/SSL_2.html`.

[125] M. J. Hinek, M. K. Low, and E. Teske. On some attacks on multi-prime RSA. In *Proc. Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002*, pages 385–404., St. John's, Canada, 2002.

[126] Claudia Hirsch. Personal communication, 2005.

[127] Ursula Holmström. User-centered design of security software. `http://www.tml.tkk.fi/Research/TeSSA/Papers/Holmstrom/`, 1999.

[128] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, 2002.

[129] Brian Hunter and Bartol Filipovíc. Enabling PKI Services for Thin-Clients. *Datenschutz und Datensicherheit*, 26, 2002.

[130] C. Irvine and T. Levin. Towards a taxonomy and costing method for security services. In *Proc. 15th Annual Computer Security Applications Conference*. IEEE, 1999.

[131] ISO. *Standard no. 9241 – part 10: Guidelines for dialog design.*

[132] Uwe Jendricke. *Sichere Kommunikation zum Schutz der Privatsphäre durch Identitätsmanagement.* PhD thesis, Universität Freiburg, 2003.

[133] Uwe Jendricke and Daniela Gerd tom Markotten. Usability meets security – the identity-manager as your personal security assistant for the internet. In *Proceedings 16th Annual Computer Security Applications Conference*, pages 344–353, 2000.

[134] M. Joye and R. Pinch. Cheating in Split-Knowledge RSA Parameter Generation. In *Proc. Workshop on Coding and Cryptography*, pages 157–163, Paris, France, 1999.

[135] David Kahn. *The Codebreakers.* Scribner, 2nd edition, 1996.

[136] Clare-Marie Karat. Iterative usability testing of a security application. In *Proceedings of the Human Factors Society 33rd Annual Meeting*, 1989.

[137] K. Karvonen. Designing trust for a universal audience: a multicultural study on the formation of trust in the Internet in the Nordic Countries. In *HCI*, pages 1078–1082, 2001.

[138] K. Karvonen and L. Cardholm. Cultures of trust: A cross-cultural study on the formation of trust in an electronic environment. In *Proceedings of the 3rd Nordic Workshop on Security (NordSec 2000)*, 2000.

[139] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *EUROCRYPT*, pages 475–494, 2001.

[140] Marko Keller and Sebastian Ries. Implementierung zweier Protokolle von Gilboa zur verteilten RSA-Schlüsselerzeugung für zwei Parteien (Praktikumsbericht). 2004.

[141] Kerckhoff. La cryptographie militaire. *Journal des sciences militaires*, IX, 1883.

[142] kes/Microsoft. Lagebericht zur informationssicherheit. *kes*, 4/5, 2004.

[143] R. Khare and S. Lawrence. Upgrading to TLS Within HTTP/1.1. RFC 2817, May 2000.

[144] Judith Klink and Tobias Straub. Rechtliche und technische Entwicklungen digitaler Signaturen. In *Proceedings D-A-CH Security*, 2005.

[145] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). RFC 1510, September 1993.

[146] Loren M. Kohnfelder. Toward a practical public-key cryptosystem. Master's thesis, MIT Department of Electrical Engineering, 1978.

[147] D. Kolb. *Experiential Learning*. Prentice Hall, Englewood Cliffs, NJ, 1984.

[148] Olga Kornievskaia, Peter Honeyman, Bill Doster, and Kevin Coffman. Kerberized Credential Translation: A Solution to Web Access Control. In *Proc. 10th USENIX Security Symposium*, pages 235–250, May 2001.

[149] D. Kristol and L. Montulli. HTTP State Management Mechanism. RFC 2965, October 2000.

[150] Martin Laabs, Matthias Merz, and Jan Wunderlich. Benutzbarkeitsevaluation von Trustcentersoftware: Microsoft Windows 2003 Server CA und Entrust Authority 7.0. Technical report, TR, 2005.

[151] Martin Laabs, Matthias Merz, Jan Wunderlich, and Tobias Straub. Benutzbarkeitsevaluation von Trustcenter-Software am Beispiel der Windows 2003 Server CA. In *Proc. D-A-CH Security*, 2005.

[152] A. K. Lenstra. Unbelievable Security: Matching AES security using public key systems. In *Proc. Advances in Cryptology – ASIACRYPT 2001*, pages 67–86, Gold Coast, Australia, 2001.

[153] R. Levien, L. McCarthy, and M. Blaze. Transparent Internet e-mail security (draft version). Technical report, AT&T Laboratories, Murray Hill, 1996.

[154] Marcus Lippert. personal communication, 2005.

[155] Silvia Lippmann and Heiko Rossnagel. Geschäftsmodelle für signaturgesetzkonforme trust center. In *Proc. Wirtschaftsinformatik*, 2005.

[156] Avivah Litan. Criminals Exploit Consumer Bank Account and ATM System Weaknesses. `http://www.gartner.com/AnalystBiography?authorId=12030`, 2005.

[157] A. Luotonen. Tunneling TCP Based Protocols Through Web Proxy Servers. `http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt`, 1998.

[158] P. MacKenzie and M. K. Reiter. Networked Cryptographic Devices Resilient to Capture. *International Journal of Information Security*, 2(1), 2003.

[159] Philip MacKenzie and Michael K. Reiter. Two-party generation of DSA signatures. *Lecture Notes in Computer Science*, 2139, 2001.

[160] M. Malkin, T. Wu, and D. Boneh. Experimenting with Shared Generation of RSA keys. In *Proc. 1999 Network and Distributed System Security Symposium*, pages 43–56, San Diego, USA, 1999.

[161] Sönke Maseberg. Fail-Safe-Konzept für PKIs. *Datenschutz und Datensicherheit*, 27(2):79–83, 2003.

[162] Ueli Maurer. Modelling a public-key infrastructure. *Proc. European Symposium on Research in Computer Security*, 1996.

[163] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[164] Microsoft. Errorneous verisign-issued digital certificates pose spoofing hazard. `http://www.microsoft.com/technet/security/bulletin/MS01-017.asp`, 2001.

[165] Microsoft Corporation. Microsoft NTLM. *Microsoft Developer Network Library*, June 2004. URL: `http://msdn.microsoft.com/library/en-us/secauthn/security/microsoft_ntlm.asp`.

[166] David L. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305, March, 1992.

[167] Muñoz, Jose L., Jordi Forné, Oscar Esparza, and Miguel Soriano. CERVANTES – A Certificate Validation Test-Bed. In *Proc. EuroPKI*, 2004.

[168] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. Online Certificate Status Protocol – OCSP. RFC 2560, 1999.

[169] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In *Proc. ACM Conference on Computer and Communications Security*, pages 59–66, San Francisco, USA, 1998.

[170] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. 31st STOC*, pages 245–254, 1999.

[171] Netcraft. Certificate Trustworthiness. `http://survey.netcraft.com/surveys/analysis/https/2004/Jan/trust.html`, 2004.

[172] Netscape Communications Corp. *Navigator Proxy Auto-Config File Format*, March 1996. URL: `http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html`.

[173] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

[174] Jakob Nielsen. Usability inspection methods. In *Proc. CHI*, pages 413–414, 1994.

[175] Jakob Nielsen. User education is not the answer to security problems. Alertbox, October 2004.

[176] Jakob Nielsen and Robert L. Mack. *Usability Inspection Methods*. John Wiley and Sons, 1994.

[177] NIST. Information Technology Security Training Requirements: A Role- and Performance-Based Model. Technical report, NIST Special Publication 800-16, 1998.

[178] L. Nitschke. Interactive Key Generation Protocols. In *Proc. WartaCrypt*, 2004.

[179] Donald A. Norman. *The Design of Everyday Things*. Basic Books, 2nd edition, 2002.

[180] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. EUROCRYPT*, pages 223–238, 1999.

[181] EU Parliament. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. `http://europa.eu.int/eur-lex/lex/LexUriServ/LexUriServ.do?uri=CELEX:31999L0093:EN:HTML`, January 2000.

[182] A. Pashalidis and C. J. Mitchell. A Taxonomy of Single Sign-On Systems. In *Proc. Information Security and Privacy – 8th Australasian Conference, LNCS 2727*, pages 249–264. Springer-Verlag, 2003.

[183] N. Pohlmann. Die virtuelle Poststelle. In *IT-Sicherheit im verteilten Chaos*, 2003.

[184] Norbert Pohlmann. Anti-Spam Technologie. In *Proc. D-A-CH Security*, 2004.

[185] G. Poupard and J. Stern. Generation of Shared RSA Keys by Two Parties. In *Proc. Advances in Cryptology – ASIACRYPT '98*, pages 11–24, Beijing, China, 1998.

[186] D. Raggett, A. Le Hors, and I. Jacobs. HTML 4.01 Specification W3C Recommendation. 1997. URL: `http://www.w3.org/TR/html4/`.

[187] Rainbow Technologies. Password Survey Results. `http://mktg.rainbow.com/mk/get/pwsurvey03`, 2003.

[188] Kai Rannenberg, Andreas Pfitzmann, and Günter Müller. Sicherheit, insbesondere mehrseitige IT-Sicherheit. In *Mehrseitige Sicherheit in der Kommunikationstechnik*. Addison-Wesley, 1997.

[189] Man Young Rhee. *Internet Security*. John Wiley and Sons, 2003. ISBN 0-470-85285-2.

[190] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. *Lecture Notes in Computer Science*, 2248:552–565, 2001.

[191] Ronald R. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Comm. of the ACM*, 21(2):120–126, 1978.

[192] Volker Roth, Kai Richter, and Rene Freidinger. A PIN-Entry Method Resilient Against Shoulder Surfing. In *Proceedings CCS'04*, Washington, DC, USA, 2004. ACM.

[193] Volker Roth, Tobias Straub, and Kai Richter. Security and usability engineering with particular attention to electronic mail. *Int. J. Hum.-Comput. Stud.*, 63(1-2):51–73, 2005.

[194] RSA Laboratories. PKCS#11: Cryptographic Token Interface Standard. Standard. URL: `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/`.

[195] RSA Laboratories. PKCS#8: Private-Key Information Syntax Standard. Standard. URL: `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-8/`.

[196] RSA Laboratories. PKCS#12 v1.0: Personal Information Exchange Syntax. Standard, June 1999. URL: `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-12/`.

[197] RSA Laboratories. PKCS #1 v2.1: RSA Cryptography Standard. Standard, 2002. URL: `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/`.

[198] K. Rudolph, G. Warshawsky, and L. Numkin. *Computer Security Handbook*, chapter Security Awareness. 2001.

[199] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Communications of the ACM*, 17(7), 1974.

[200] Ravi Sandhu. Good-enough security, toward a pragmatic business-driven discipline. *IEEE Internet Computing*, 7(1):66–68, 2003.

[201] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 522–533, New York, NY, USA, 1994. ACM Press.

[202] M. A. Sasse. Computer security: Anatomy of a usability disaster, and a plan for recovery. In *CHI 2003, Workshop on Human-Computer Interaction and Security Systems*, Fort Lauderdale, USA, 2003.

[203] M. Angela Sasse. Usability and trust in information systems. Cyber Trust & Crime Prevention Project, 2004.

[204] M.A. Sasse, S. Brostoff, and D. Weirich. Transforming the 'weakest link' – a human/computer interaction approach to usable and effective security. *BT Technology J.*, 19(3), 6 2001.

[205] B. Schneier. *Applied Cryptography*. John Wiley and Sons, 1996.

[206] B. Schneier. *Secrets and Lies*. John Wiley and Sons, 2000.

[207] Bruce Schneier and Chris Hall. An improved e-mail security protocol. Technical report, Counterpane Systems, Minneapolis, 1997.

[208] M. Schumacher, U. Roedig, and M.-L. Moschgath. *Hacker Contest. Sicherheitsprobleme, Lösungen, Beispiele*. Springer, 2002.

[209] H. Seemann. Pragmatic Solutions to Make E-mail Security Work. In *Proc. Information Security Solutions Europe*, 2003.

[210] A. Shamir. Identity Based Cryptosystems and Signature Schemes. In *Proc. Advances in Cryptology – Cryto'84*, number 0196 in LNCS. Springer, 1984.

[211] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[212] R. Shirey. Internet Security Glossary. RFC 2828, 5 2000.

[213] Axel Sikora and Tobias Straub. Cryptography on Embedded Systems. In *Proceedings Embedded World*, 2005.

[214] M. T. Siponen. Five dimensions of information security awareness. *Computers and Society*, 6:24–29, 2001.

[215] D. K. Smetters and R. E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. In *New Security Paradigms Workshop '02*. ACM, 2002.

[216] Shannon Sofield, Dave Nielsen, and Dave Burchell. *PayPal Hacks*. O'Reilly, 2004.

[217] E. Spyropoulou, T. Levin, and C. Irvine. Calculating costs for quality of security service. In *Proc. 16th Computer Security Applications Conference*. IEEE, 2000.

[218] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proc. 7th International Security Protocols Workshop*, pages 172–194, 1999.

[219] William Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 2nd edition, 1999.

[220] W. Stangl. Lernstile – Theoretische Modelle. `http://www.stangltaller.at`, 2005.

[221] J. G. Steiner, C. Neumann, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proc. USENIX Winter Conference*, pages 191–202, 1988.

[222] J. P. Stern. A New and Efficient All-Or-Nothing Disclosure of Secrets Protocol. In *Proc. Advances in Cryptology – ASIACRYPT '98*, pages 357–371, Beijing, China, 1998.

[223] J.E. Stice. Using Kolb's Learning Cycle to Improve Student Learning. *Engr. Education*, 77:291–296, 1987.

[224] Douglas R. Stinson. *Cryptography: Theory and Practice*. CDC Press, 1995.

[225] Tobias Straub. Key Fingerprint Verification from an HCISec Perspective. (in preparation).

[226] Tobias Straub. Efficient Two Party Multi-Prime RSA Key Generation. In *Proc. IASTED International Conference on Communication, Network, and Information Security*, 2003.

[227] Tobias Straub. A Method for Strengthening Certificate Enrollment. In *Proc. WartaCrypt*, 2004.

[228] Tobias Straub. On Usability Issues of PKI-enabled Applications. In *Proc. 3rd Conference on Communications, Internet and Information Technology (CIIT)*, St. Thomas, Virgin Islands, USA, 2004.

[229] Tobias Straub. PKI-Outsourcing: Vertrauen ist gut, Kryptografie ist besser. In *Proc. 11th DFN-CERT Workshop Sicherheit in vernetzten Systemen*, Hamburg, Germany, 2004.

[230] Tobias Straub. Spezifikation von X.509-Zertifikatsprofilen unter dem Gesichtspunkt Benutzbarkeit. In *Proc. D-A-CH Security*, 2004.

[231] Tobias Straub. System zur Absicherung der Erstellung von Public-Key-Zertifikaten. Gebrauchsmusteranmeldung 20 2004 012 201.4 (pending), 2004.

[232] Tobias Straub. TISP: Expertenzertifikat für IT-Sicherheit. In *Presentation at the Workshop "Motivation und Schulung zur IT-Sicherheit"*. GI-Fachgruppe Security Management, 10 2005.

[233] Tobias Straub and Harald Baier. A Framework for Evaluating the Usability and the Utility of PKI-enabled Applications. In *Proc. EuroPKI*, number 3093 in Lecture Notes in Computer Science, pages 112–125. Springer, 2004.

[234] Tobias Straub, Matthias Fleck, Ralf Grewe, and Oliver Lenze. SecMGW – An Open-Source Enterprise Gateway for Secure E-Mail. In *Proc. Information Security Solutions Europe*, Berlin, 2004.

[235] Tobias Straub, Thilo-Alexander Ginkel, and Johannes Buchmann. A Multipurpose Delegation Proxy for WWW Credentials. In *Proc. Second European PKI Workshop (EuroPKI)*, volume 3545 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2005.

[236] Tobias Straub, Manuel Hartl, and Markus Ruppert. Digitale Reisepässe in Deutschland – Prozesse und Sicherheits-Infrastruktur. In *Proc. SICHERHEIT*, 2006. (accepted for publication).

[237] Tobias Straub and Andreas Heinemann. An anonymous bonus point system for mobile commerce based on word-of-mouth recommendation. In *Proc. SAC*, pages 766–773, 2004.

[238] Bob Sullivan. Consumers still falling for phish – Fake e-mails fool users 28 percent of the time, study finds. *MSNBC News*, July 2004.

[239] Bob Sullivan. Survey: 2 million bank accounts robbed – Criminals taking advantage of online banking, Gartner says. http://www.itfacts.biz/, June 2004.

[240] Ruth Thomas. National campaign launched to help UK get safe online. `http://www.nhtcu.org`, 10 2005.

[241] D. Thomsen and M. Denz. Incremental assurance for multilevel applications. In *Proc. 13th Annual Computer Security Applications Conference*. IEEE, 1997.

[242] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) – Proxy Certificate Profile. RFC 3820, June 2004.

[243] Christian Valentin. Ein Verteilter Zeitstempeldienst basierend auf RSA in Java. Master's thesis, TU Darmstadt, 2001.

[244] VeriSign. Fraud Detected in Authenticode Code Signing Certificates. `http://www.verisign.com/support/advisories/authenticodefraud.html`, 2001.

[245] J. Voßbein and R. Voßbein. KES/KPMG-Sicherheitsstudie: Lagebericht zur ITSicherheit. *kes*, (3,4), 2002. available online `http://www.kes.info`.

[246] Viktor L. Voydock and Stephen T. Kent. Security mechanisms in high-level network protocols. *ACM Computing Surveys*, 15(2):135–170, 1983.

[247] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, 2004. `http://eprint.iacr.org/`.

[248] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In *Proc. CRYPTO*, pages 17–36, 2005.

[249] G. Wearden. Few takers for security outsourcing. *ZDNet UK*. `http://news.zdnet.co.uk/internet/security/0,39020375,2133349,00.htm`.

[250] Dirk Weirich and Martina Angela Sasse. Persuasive password security. In *CHI'01 extended abstracts on Human factors in computing systems*, pages 139–140, New York, NY, USA, 2001. ACM Press.

[251] A. Whitten. *Making Security Usable*. PhD thesis, Carnegie Mellon University, 2004.

[252] A. Whitten and J.D. Tygar. Safe Staging for Computer Security. In *Proc. of the Workshop on Human-Computer Interaction and Security Systems*, 2003.

[253] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *Proc. 8th USENIX Security Symposium*, 1999.

[254] Bruno Wiederkehr. IT Security Awareness Programme. *Information Systems Control*, 3, 2003.

[255] R. N. Wright and S. Spalding. Experimental Performance of Shared RSA Modulus Generation. *Algorithmic*, 33(1):89–103, 2002.

[256] T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant applications. In *Proc. 8th USENIX Security Symposium*, pages 79–92, Washington, USA, 1999.

[257] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2004.

[258] Ka-Ping Yee. User interaction design for secure systems. In *Proc. 4th International Conference Information and Communications Security*, volume 2513 of *LNCS*, pages 278–290. Springer, 2002.

[259] Ka-Ping Yee. Aligning security and usability. *IEEE Security & Privacy*, pages 48–55, 2004.

[260] Mary Ellen Zurko and Richard T. Simon. User-centered security. In *Proceedings of the UCLA conference on New security paradigms workshops*, 1996.