Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Patrik Bichsel

Theft and Misuse Protection for Anonymous Credentials

Master Thesis MA-2007-42
June 2007 to November 2007

Tutors: Dr. Jan Camenisch
Thomas Gross
Dieter Sommer
Co-Tutor: Bernhard Tellenbach
Supervisor: Prof. Dr. Bernhard Plattner

Acknowledgment

It has been a very interesting and challenging task to get to know the state-of-the-art technology for privacy protection. Just as well, I enjoyed the technological analysis of already proposed misuse protection mechanisms.

Most special thanks to Dieter Sommer, who always found a competent answer to my questions. Even so, I want to thank Thomas Gross, Jan Camenisch, Thomas Heydt-Benjamin and last but not least, Michael Baentsch for helping me through the daily trouble that I faced. Also, I would like to thank the IBM Zurich Research Laboratory for providing the resources and giving me the opportunity to work in this interesting field of research. Thanks also to Bernhard Tellenbach who was willing to support me while I have been working on the thesis. Additionally, I want to express my gratitude to the head of the Communication Systems Group, Prof. Plattner, for taking the patronage of this thesis.

Zurich, November 31, 2007

Patrik Bichsel

Abstract

Today, many service providing organisations require their users to authenticate before enabling them to use their services. A widely used system for this purpose is the so called *username-password system*. The most important advantages of this system are the ease of use, the experience in implementing and the acceptance of users. However, this system has some severe disadvantages. The first lies in the fact that many users have a vast amount of accounts. As an example there are such accounts for email, online auctions, gaming and online banking. Trying to reduce the entropy they need to store in their brain, they tend to choose the same username-password combination for authentication with different organisations. This is particularly relevant if services with different security levels, such as a bank account and an email account, are accessed with the same authentication information.

Federated identity management allows a user to acquire a credential with certified attributes. He only needs to store the credential in a safe place and can authenticate using the credential. The authentication process transparently uses public key cryptography and therefore offers much better security compared to the username-password approach. Another advantage of federated identity management is the possibility to performing authentication across borders of different domains (e.g., companies). An anonymous credential system, such as *idemix*, can be extended such that it implements the concept of identity federation. It uses not only credentials with certified attributes but guarantees that the identity of each user will be protected. The privacy protection reaches even further, as the actions of a user with a credential are not linkable. As an example, a user can authenticate several times with one company using the same credential and still the organisation will not gain any information apart from the fact that an eligible user has accessed its system. Such a far-reaching privacy protection has disadvantages, too. Credentials are issued as digital information which enables users to share them arbitrarily. Moreover, the credential can be duplicated by malicious software without the owner even noticing. In other words, anonymous credential systems do neither prevent users from sharing their credentials nor can they detect if several copies of one credential are used extensively by different users. This may hinder the large-scale adoption of anonymous credential systems.

In this thesis, we will first present an overview of several techniques to cope with credential misuse, such as sharing or theft, in anonymous credential systems. A comparison of the techniques will be the basis for the choice of an effective misuse prevention mechanism. We will then describe how two concrete techniques — K-show credentials and hardware-bound credentials — have been implemented into the *idemix* system. The result is a system offering strong privacy combined with effective sharing and theft prevention.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays more and more services become accessible over networks. The services are provided by organisations, which want to make it accessible for a number of users. While these services seem to have many benefits for their users, at the same time they prove to introduce a severe disadvantage. This drawback is associated with the authentication process required by the organisations. Hereby, not the actual authentication process causes problem, the trouble arise due to the system being used for authenticating users. Today, the most used system for this purpose is a so called username-password approach. Its ease of use is a reason for the wide spread implementation, but unfortunately the system offers many attack vectors.

The need for authentication is explained with the protection of the assets of the organisation. Additionally, it is a useful technique to separate the assets of the system users, for example separate the e-mail accounts of different users. However, during the setup of a username-password combination, the users often are urged to provide much information to the organisation and thereby reduce their own privacy. Not many users have realised yet, to what extent this data could be used. For example, the payment transaction to a travel agency indicates an easy target for a burglary. A completely different misue of private information is the use of acquired information to send specifically targeted advertisements to the user. The last example is already a lucrative business.

An anonymous credential system, such as *idemix*, is an authentication system which offers far-reaching privacy protection for the users. It offers several other advantages over username-password system, too. An important merit is the use of certified attributes, which strengthens the assurance of information provided to the organisations. A bit contra-intuitive is the fact that the attributes do not need to be revealed to the organisation and still the assertion given to the organisations is stronger compared to the username-password approach. The primitive that is used to achieve this goal are so called zero-knowledge proofs. The setup of this system leads to very strong privacy protection of the user.

Another benefit of *idemix* is the possibility to extend it to a federate identity management system. Federated identity management allows a user to authenticate across the borders of different domains. For example, let us assume that an organisation O_1 provides the IT infrastructure to another organisation O_2. If an employee of O_1 wants to access a computer room at the location of organisation O_2, he nowadays needs to authenticate to O_2. However, as O_1 is managing those resources by order of O_2, the access should be granted. Federated identity management makes it possible for the employee to authenticate with O_1 which authenticates to O_2 on his behalf. Clearly, the usability of many systems could be enhanced using this technique.

To issue certified attributes to a user, the *idemix* system uses so called credentials. As they contain certified attributes, they are also denoted as certificates. The credentials are issued to the user by an identity providing organisation and contain information which is signed by the identity providing organisation. The attained attributes can be used to acquire more credentials or to authenticate at some service providing organisation.

As credentials are digital information, they are easy to copy. This is a problem as a user, for example, can copy his credentials and share them with another user. Such practices are clearly a misuse of the credential system. *Idemix* current lacks the ability to cope with duplicate credentials. In particular, there are no possibilities to detect or even prevent such misbehaviour. Especially, the anonymity of the system seems to be in conflict with a mechanism to prevent misuse. However, there have been different approaches proposed which limit the misuse possibilities of users in the system. Some accomplish this objective by limiting the incentive of the user to copy his credentials. Others limit the number of usages of a credential in a certain time interval to reduce the number of copies that can be used simultaneously without being detected. Anyhow, none of the above mentioned methods can prevent the occurrence of credential duplicates. This results as the discussed approaches are all cryptography-based.

A completely different angle of approaching the problem is the following. The idea is to transform credentials form digital information into physical objects. Clearly, such a transformation cannot be achieved. Nevertheless, the enclosure of essential information in a tamper resistant object attains the envisioned goal. A credential is not mere digital information anymore. Thus, the ability to freely copy the credential is eliminated. Even though the introduction of tamper-resistant, i.e., secure hardware reduces the problem of duplicate credentials to a certain extent, it is not the ultimate solution to misuse protection. This results from the fact that even data residing in a tamper-resistant device can be extracted. Hence, there needs to be another mechanism to limit the possibilities after a secure hardware device has been broken. We propose to use the K-show credentials as described by Camenisch et al. in [3] in addition to the hardware-based protection mechanism. Consequently, we achieve an anonymous credential system with an effective misuse protection. The *idemix* system is thereby considered ready to be used in real world scenarios and its spread is supported by including *idemix* into the open-source Higgins Trust Framework[1].

In addition to the choice of an effective misuse protection mechanism, we provide a prototype implementation of this mechanism. It is embedded into the already implemented *idemix* system. The computation time of the prototype is measured and the prototype implementation is found to perform poorly. The reason of which is the arithmetic which is running in the application layer. The necessity of this measure is discussed in broader detail. As a reaction of the poor performance of the implemented system, we provide an exploration of expected runtimes with the use of the embedded cryptographic coprocessor.

Let us give a brief overview of the structure of the thesis. After the problem statement provided as Chapter 2, we discuss the theoretical basics which are necessary for the understanding of the thesis. Afterwards, Chapter 4 focuses on credential systems by giving an introduction on the basic mechanisms as well as by comparison with well-known authentication systems. This chapter also reasons about the choice of a credential system as authentication system for future applications. Chapter 5 provides an overview over the different techniques that can be used for misuse protection in anonymous credential systems. The aim of the succeding chapter is to highlight in extensive detail the implemented misuse protection mechanism. In addition the *idemix* system is briefly described to allow the understanding of the implementation part. Subsequently, Chapter 7 talks about the extensions that effectively needed to be made to the *idemix* system. In addition to that, it describes all other implementations that were necessary, i.e., especially the implementation of a arbitrary position arithmetic for a Java card is motivated. Thereafter in Chapter 8 the findings are summarised and an outlook on future work is given.

[1]Refer to `http://www.eclipse.org/higgins/` for more details.

Chapter 2

Problem Definition

2.1 Theft and Misuse Protection for Anonymous Credentials

Anonymous Credential Systems allow a user to authenticate in Web transactions by proving certified attributes about him or herself while maintaining strong privacy protection. Anonymous Credential Systems are capable of even establishing full unlinkability of transactions modulo the linking enabled by the attributes disclosed.

This capability bears the risk that anonymous credentials may be misused by malicious users, e.g., by massive sharing of the credential in peer-to-peer settings, or stolen by for instance malware, where the anonymity properties foil the possibility to identify stolen credentials or thieves. Therefore, reliable mechanisms for theft and misuse prevention are even more important for Anonymous Credential Systems than for normal username/password or public key mechanisms.

Research envisions two major solutions for these challenges. (a) The anonymous credentials may be tied by cryptographic means to a hardware device and cannot be used without the device being present in the transaction. (b) The anonymous credential may be extended by compact e-cash primitives to prevent massive multiple use of the credential: if the credential is for instance invoked more than ten times a day, the identity of the owner is automatically revealed by the multiple-spending prevention of the e-cash.

This research project is to explore the theft and misuse protection capabilities for the Identity Mixer, an Anonymous Credential System based on Camenisch-Lysyanskaya signatures. The project involves current research in cryptography as well as system architecture and implementation for security systems. The project is aligned with the open-source user-centric identity management project Higgins.

Qualifications:

- Understanding of cryptography and security
- Design for security systems
- Programming in Java and software engineering methods

The project consists of the following deliverables:

- Written elaboration on theft and misuse mechanisms for Anonymous Credential Systems to be published as research report or scientific paper.
- Design for the integration of theft and misuse protection mechanisms into the Higgins Trust Framework client and server infrastructure.

- Implementation of the mechanisms for the Higgins Trust Framework and its Identity Mixer plug-in. Requirements: the implementation must be demoable, contributable to open-source, and reusable.

Chapter 3

Theoretical Background

In this chapter we provide cryptographic basics as well as definitions of some fundamental terms and concepts. The description of the concepts is provided as far as necessary for the understanding of the thesis. Most of the concepts are referred to within the thesis. Some descriptions, however, are only described as they are necessary for a further understanding of the *idemix*[1] system. Therefore this chapter is a mere reference in case more detail on the used concepts is needed.

3.1 Fundamental Terms

Identification and authentication are terms that will be used often throughout the thesis. We want to provide an intuition more than a definition of the terms to prevent any misunderstanding.

Authentication is the process of determining whether someone is who he claims to be. It is carried out between two parties called the *prover* and the *verifier*. If the entity acting as prover can successfully convince the verifier that he is who he claims to be, he is authenticated.

Identification is the process of verifying the identity of someone. As in the case of authentication it is carried out between a *prover* and a *verifier*.

In other words we could describe identification as a process closely related to authentication. The only difference being that after successful identification the verifier is convinced to know the real name of the prover. In the authentication process the condition of the verifier must not be the knowledge of the identity. The prerequisite can for example be a paid subscription. In case of a successful authentication we consequently never assume that the verifier can link the prover to a value such that the verifier will recognise when the same prover returns.

3.2 Mathematical Preliminaries

The different cryptographic terms we use, need some explanation. Some properties of described systems hold under certain assumptions. Those assumptions are defined here. Accordingly, the following sections provide the cryptographic insight as far as needed for the understanding of the thesis.

3.2.1 Cryptographic Assumptions

First of all, two definitions are needed to be able to discuss a number of assumptions.

Definition 1 (RSA modulus)**.** An RSA modulus n is the product of two primes p and q such that $n = p \cdot q$. Let us name an RSA modulus built of two safe primes a *special RSA modulus*.

[1]Refer to Section 6.3 on page 53 for a brief introduction.

Definition 2 (Safe prime)**.** A safe prime is a prime number p which follows from another prime number p' (called *Sophie German* prime) by computing $p = 2 \cdot p' + 1$.

Strong RSA Assumption

This assumption states that it is infeasible to determine $e > 1$ and $x \in \mathbb{Z}_n^*$ when provided with an RSA modulus n and an element $y \in \mathbb{Z}_n^*$ that solve the equation $y = x^e \pmod{n}$. The modulus n needs to be a special RSA modulus. [3]

Interval RSA Assumption

Given an exponent e and an RSA modulus n it is infeasible to find a pair fulfilling $y = x^e \pmod{n}$. [1]

The last problem is easier to solve than the RSA problem which is intuitive as not the exponent to a given base needs to be found. This yields an additional degree of freedom and makes the assumption stronger compared to the RSA assumption.

The formulation of the decisional Diffie-Hellman assumption requires some additional definitions.

Definition 3 (Group)**.** A Group G with an operator $\circ$ is denoted as $(G, \circ)$ and fulfils these properties:

- *Neutral Element*: $\exists$ a neutral element denoted as $\mathbf{1}$ with $\mathbf{1} \circ\ a = a \circ \mathbf{1} \in G$, $\forall a \in G$
- *Inverse Element*: $\exists$ an inverse element i for which $a \circ\ i = i \circ\ a = \mathbf{1}$, $\forall a \in G$
- *Closure*: for all $a, b \in G$ the element $a \circ\ b$ is also in G
- *Associativity*: $a \circ\ (b \circ\ c) = (a \circ\ b) \circ\ c$ holds for all $a, b, c \in G$

Lemma 3.2.1 (Cyclic Group)**.** *A cyclic group* $(G^*, \circ)$ *is a group where the elements can be generated by the computation of the group operation* $\circ$ *on any element of the group.*

Definition 4 (Multiplicative Group)**.** Let n be a number in the group denoted by $\mathbb{Z}$. Additionally let the set $\mathcal{I}$ be the numbers $i \in \mathbb{Z}$ for which $\gcd(i, n) = 1$ holds. Then $\mathbb{Z}_n^*$ is the multiplicative group consisting of the set $\mathcal{I}$.

Definition 5 (Generator)**.** The element g which is able to generate a cyclic group $(G^*, \circ)$ is called a generator of this specific group G^*.

Discrete Logarithm Problem

Let g, h be elements of the group $(G, *)$ and x an arbitrary number in $\mathbb{Z}$. The discrete logarithm problem states that for a group with an appropriately chosen order $|G| = n$ it is difficult to find the number x solving the equation:

$$g^x = h \pmod{n}$$

Although the discrete logarithm problem applies to all multiplicative groups as defined in Def. 3 for cryptographic purposes mostly groups of the form $\mathbb{Z}_n^*$ as defined in Def. 4 are used. After the definitions of a group, a generator and a cyclic group, we can define the decisional Diffie-Hellman assumption.

Decisional Diffie-Hellman Assumption

The DDH assumption states that in a cyclic group G of order $|G| = q$ with a randomly chosen generator g, the element formed as g^{ab} is independent of g, g^a and g^b. Stated differently the following holds:

Theorem 3.2.2. *If $G = \langle g \rangle$ is a cyclic group of order q and g is a generator thereof which is chosen at random. Additionally $a, b, c \in \{0, \ldots, q-1\}$ are all chosen randomly. If the DDH assumption holds then the following terms are distributionally equal.*

$$\left(g, g^a, g^b, g^{ab}\right) =_d \left(g, g^a, g^b, g^c\right)$$

This assumption is far stronger than the computational Diffie-Hellman assumption and the discrete logarithm problem. For the latter it is obvious that by solving the discrete logarithm problem the attacker would be able to compute $a = \log_g g^a$ and consequently $g^{ab} = (g^b)^a$ which would solve the DDH as well.

3.2.2 The Random Oracle Model

A strict definition of the random oracle model is not needed here. However, an intuitive description serves for the understanding of the basic concept. The random oracle provides upon input i a truly random output o which is uniformly distributed in its output domain. To make the model practical, it answers always with output $o(i)$ when challenged with the same input i.

In particular, the random oracle model is used to model cryptographic hash functions. Especially, extensive requirements on the randomness of the output of the hash function with regards to a proof make the random oracle model useful. In such a case, a property can be proven under the random oracle model. The proof is holding if it could only be broken by requiring impossible behaviour from the random oracle, i.e., breaking a problem which is believed to be hard.

3.2.3 Commitment Scheme

The aim of a commitment scheme is for a party A to proof to its communication partner B that it knows some value x. A can show x to B after having committed to it, where B is able to verify the correctness of the given x w.r.t. the commitment. Two fundamental properties of a commitment scheme can be extracted as stated in [14]. One is the *binding property* which means that A needs to choose a value and cannot change the chosen value after he committed to it. The other is the *hiding property* which implies that B cannot extract the value x from the commitment. It requires A to provide B with the value x if B should get to know the value.

The binding and hiding property are mutually exclusive which leaves room for optimisation. The Pedersen scheme for example, achieves perfect hiding but only a computationally secure binding. A perfectly hiding but only computationally binding scheme is shown after the Pedersen scheme.

Example 3.2.1. If a party wants to commit to a value $x \in \mathbb{Z}_q$, it can issue a Pedersen commitment. This scheme suggests the computation of the commitment with $C = g^x\ h^r$ where $G = \langle g \rangle = \langle h \rangle$, i.e. g and h are generators of group G. Furthermore, $r \in \mathbb{Z}_q$ is a randomly chosen value to blind the committed value x. The two values (x, r) are called commitment opening information as they enable B to verify the commitment C. As mentioned above, the Pedersen commitment scheme is perfectly hiding.

Example 3.2.2. Given is a group with $G = \langle g \rangle = \langle h \rangle$ and x as described in Example 3.2.1. A scheme that is perfectly binding but only computationally hiding is achieved by computing the commitment as $C = (g^x h^r, h^x)$. Again the commitment is opened by supplying B with the values x and r. [13]

3.3 Zero Knowledge Proof of Knowledge

A zero-knowledge proof has the goal of proving knowledge of a statement without revealing the statement itself to the other party. For example, an entity is willing to prove knowledge of a password. Apparently, it does not want its communication partner to get to know the password. In this case, the entity could issue a zero-knowledge proof of the password and send it to its communication partner. The latter would learn that the entity knows the password without getting to know the password itself.

An introductory example given by Quisquater and Guillou [21] of a zero-knowledge proof of knowledge uses a cave with shape similar to the one depicted in Fig. 3.1. The prover called *Peggy* wants to prove to the verifier *Victor* that she knows a secret. The secret can be used to open the door in the middle of the cave. However, she does not want *Victor* to get to know the secret.

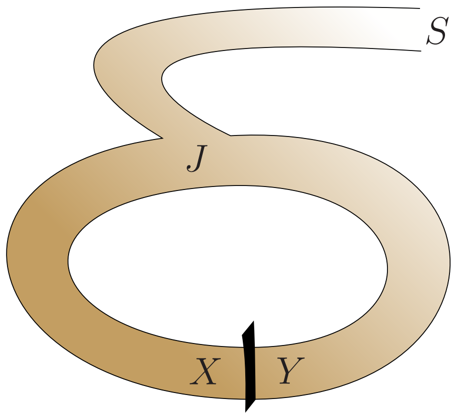

Figure 3.1. Visualisation of the zero knowledge paradigm

A scheme which allows *Peggy* to achieve her goal is the following. She enters the cave and goes, starting from the location S, to either one of the dead ends X or Y. In the meanwhile, *Victor* waits at the starting point S. As soon as *Peggy* has reached her destination, *Victor* goes to junction J and tells *Peggy* where she should meet him from. If Peggy did not know the secret to open the door she would only have a chance of fifty percent of having picked the right part of the cave. Therefore, *Victor* will never be completely certain that *Peggy* really knows the secret. However, after k independent repetitions of the experiment his incertitude will have fallen to 2^{-k}.

From the given example, three properties of a zero-knowledge proof of knowledge follow immediately.

Completeness : The verifier accepts the proof always if the answers of the prover are correct and both parties are following the protocol.

Soundness : Upon a wrong answer from the prover, the verifier must always rejects if both prover and verifier are following the protocol.

Zero knowledge : The verifier does not get to know any data that might be useful in learning the secret of the prover. He only learns about the correctness of the proof which does not even imply that he will be able to convince another party that he has been proven the secret.

An interactive protocol which proofs knowledge of a discrete logarithm is given in [20] and shown below. This scheme is not considered to be efficient as interactiveness requires too much communication.

Common inputs are $\langle g\rangle + \langle\tilde{g}\rangle = G_q$ and $(h, \tilde{h}) \in\ G_q$
Prover knows $x \in \mathbb{Z}_q^*$ with $h = g^x$ and $\tilde{h} = \tilde{g}^x$

$$\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
\text{choose } r \in_R \mathbb{Z}_q^* & & \\
A = g^r, B = \tilde{g}^r & \xrightarrow{A,B} & \\
 & \xleftarrow{c} & \text{choose random } c \\
y = r + cx \pmod q & \xrightarrow{y} & \\
 & & \text{verify } g^y = A \cdot h^c \\
 & & \text{and } \tilde{g}^y = B \cdot \tilde{h}^c
\end{array}$$

The notation introduced by Camenisch and Stadler in [9] can be used to denote such a zero-knowledge proof. It allows conjunction of the arguments to be proven as seen in example 3.3.1. The notation of the above zero-knowledge proof would consequently be the following.

$$PK\{(\alpha, \beta) : y = g^\alpha h^\beta\}$$

Example 3.3.1. A user U is proving the correct building of a commitment $C = g^x h^r$ in addition to a range prove of γ and a proof of knowledge of δ. Such a prove would be denoted as

$$PK\{(\alpha, \beta, \delta, \gamma) : y = g^\alpha h^\beta \wedge z = g^\delta \wedge (a \leq \gamma \leq b)\}$$

where the allowed range for γ is the interval $[a, b]$.

Definition 6 (Quadratic Residue)**.** Let p be a number with $p \in \mathbb{Z}$. Let additionally be n a modulus.

p is said to be a quadratic residue if and only if there exists a number x for which $p = x^2 \pmod n$ holds otherwise it is named a quadratic non-residue. All quadratic residues w.r.t. a modulus n can be found by computing $i^2 \pmod n$, with $i \in \left[0, \frac{n-1}{2}\right]$.

If the zero-knowledge proof is done using the group G of quadratic residues modulo a composite n which is a product of safe primes, i.e. $G = QR_n$, then the protocol must follow special rules. The first of which is that challenge c has to be smaller than the smallest factor of the group order $|G|$. Second, the protocol performs a proof under the strong RSA assumption. Hereby the soundness needs special attention as determining if an element belongs QR_n is believed to be hard for an entity not knowing the factorisation of n. However, Brickell et al. stated in [1] that it is sufficient in most cases to execute $PK\{(\alpha) : y^2 = g^{2\alpha}\}$ instead of $PK\{(\alpha) : y = g^\alpha\}$ which solves the problem that the verifier cannot verify that g is indeed a generator of group G as $g^2 \pmod n$ clearly is.

To make the proposed scheme efficient, Fiat and Shamir have shown in [15], that the proof can be changed into a signature under the random oracle model. The so called Fiat-Shamir heuristic will be denoted as signature proof of knowledge: $SPK\{(\alpha) : y = g^\alpha\}(m)$.

3.3.1 Fiat-Shamir Protocol and Signature

The Fiat-Shamir protocol is a construction which is particularly useful for device with low computation resources such as smart cards. The scheme is an early example of a protocol using zero-knowlege proofs for authentication purposes. It has special properties and which need some assumptions.

- a trusted entity T or PKI is present
- valid users do not need to be stored
- no information is leaked when showing a signature

- factorisation of modulus n is known to T only
- a hash function $f : \mathbb{N} \mapsto [0, n[$ which leads to an output that is for any polynomial time algorithm indistinguishable from a truly random source

The assumption of a trusted entity is not desirable, however it is considered to be easier to introduce compared to a working and widely accepted PKI. The second property relieves T as well as all other parties of the burden to have a list of all users which makes the scheme extremely scalable. The protocol of showing a signature is computationally zero knowledge.

The issuing of a smart card requires the user to identify with the trusted entity T. The latter needs to execute the following steps:

1. create unique identifier $I = (userName, address, ID, securityParameters, \ldots)$
2. calculate $v_i = f(I, i)$ for small i
3. pick k values for which v_i is a quadratic residue $\pmod n$ and compute the smallest square root s_i of each $v_i^{-1} \pmod n$.
4. issue card (I, s_j), with $j \in [1, \ldots, k]$

We use a bijective indexing function $g^{-1}(i) = j$ to map the k chosen values to the interval $j \in [1, k]$. By definition there exists an inverse function g reversing this mapping. In the scheme the indices i are stored on the card which is congruent to storing the mapping function g. Note that the use of this mapping function is done for convenience purposes only.

The protocol which allows a smart card and hence a user to identify with a verifier works as follows:

1. $P \rightarrow V$: send I
2. V : generate $v_j = f(I, i)$, $i = g(j)$ with $j \in [1, k]$
3. repeat for $i \in [0, t[$
   $P \rightarrow V$: send $x_i = r_i^2$, with randomly chosen $r_i \in [0, n[$
   $P \leftarrow V$: send random binary vector $(e_{i1}, \ldots, e_{ik})$
   $P \rightarrow V$: $y_i = r_i \prod_{e_{ij}==1} s_j \pmod n$
   V: check $x_i = y_i^2 \prod_{e_{ij}==1} v_j \pmod n$

The security parameter of the scheme is the product $k \cdot t$ which is explained by the probability of V accepting the prover even though P did not know the s_j which evaluates to 2^{-kt}. The choice of k and t are consequently subject to optimisation even if the overall security of the system, i.e. kt, is fixed. The communication can be reduced by choosing a large k and reduce the number of iterations t. In the latter case the requirements on available memory are raised. Conclusively we can see that the burden can be shifted from memory intensive to communication intensive with a small t and a small k respectively.

The random vector of V prevents P from cheating. However, it can be replaced by a hash function $h(\cdot)$ to make the protocol non-interactive. This results in the following protocol between a prover P and the verifier V.

P signs message m

1. compute $x_i = r_i^2$, with randomly chosen $r_i \in [0, n[$ and $i \in [1, t]$
2. compute $h(m, x1, \ldots, x_t)$, use kt bits as e_{ij} where $i \in [1, t] \wedge j \in [1, k]$
3. compute
$$y_i = r_i \prod_{e_{ij}==1} s_j \pmod{n}, \qquad \text{for } i = 1, \ldots, t$$
4. send (I, m, e_{ij}, y_i) to V

V verifies the message m

1. compute $v_j = f(I, j)$, for $i = 1, \ldots, k$
2. compute
$$z_i = y_i^2 \prod_{e_{ij}==1} v_j \pmod{n}, \qquad \text{for } i = 1, \ldots, t$$
3. check if the first kt bits of $h(m, z_1, \ldots, z_t)$ are the same as the e_{ij}

For memory restricted device the hash function h can be chosen to be the same as hash function f without having security implications as long as the chosen hash function is considered to be secure.

A challenge lies in the fact that an attacker can verify if his signature will be accepted which makes it necessary to have an appropriate security level, i.e., to choose the product kt sufficiently large. This is convenient as even with a fixed key length k the security level can be adjusted arbitrarily.

Even though the signatures are believed to not reveal any information, their zero-knowledgeness cannot be proven. This follows from the fact that everybody can verify a signature and all the same V is unable to generate false signatures. Hence, the secret information needs to be in the signature, however, it is not extractable by V which makes the signature scheme secure. [15]

3.3.2 The Camenisch-Lysyanskaya Signature Scheme

The signature scheme introduced by Camenisch et al. in [7] is summarised here. The protocols are not described in detail as they will not be specifically addressed in the thesis. The scheme, however, is important as it allows an entity to sign a value without getting to know it, which is needed in the *idemix* system[2]. In particular, a party can commit to a value which will then be signed by the signer. In addition to this ability, the protocols allow an entity to efficiently prove knowledge of a Camensch-Lysyanskaya signature. The scheme is based on discrete-logarithm-based proofs of knowledge and shown to be secure under the strong RSA assumption.

Key generation :

- input is 1^k where k is the security parameter;
- choose a special RSA modulus $n = p \cdot q$ with length $l_n = 2k$;
- choose uniformly at random $a, b, c \in QR_n$;
- $PK = (n, a, b, c)$; $SK = p$

[2]Refer to Section 6.3 on page 53 for a brief introduction.

Message space :

- the messages $\{(m_1, \dots, m_L) : \forall m_i \in \{0,1\}^{l_m}\}$ with the length parameter l_m

Signing algorithm :

- input m_i;
- choose random prime number e with length $l_e \geq l_m + 2$;
- choose random number v with $l_v = l_n + l_m + l$ where l is a security parameter;
- compute r with $r^e = a^{m_i} b^v c \pmod{n}$;
- output $\sigma = (e, v, r)$

Verification algorithm :

- check if $r^e \equiv a^{m_i} b^v c \pmod{n}$
- check if $2^{l_e - 1} < e < 2^{l_e}$

The basic operations of the signature scheme are given above. However, we do not provide the protocols as they are described in wide detail in [7]. The protocol is used whenever a signature on a secret message is needed. As such, it is a basic operation for the credential system as proposed in [5].

3.4 Dynamic Accumulators

The intuition behind an accumulator is to combine different values into one accumulator. The combination of the values is done such that the accumulator fulfils certain requirements. For example, it needs to be possible to proof that some value is in the accumulator. Yet, this proof must fail if the value has not been used in the computation of the accumulator. Such a scheme can be used to deal with revocation. For example, in a PKI[3] environment each public key that is valid could be put into the accumulator. Instead of revoking the public key, the value could be taken out of the accumulator. If now each user has not only to prove possession of the secret key but also that his key is in the accumulator, a revocation scheme would be attained. As the example shows, an important operation for accumulator schemes is the addition of values to the accumulator. Similarly, the revocation of values contained in the accumulator is just as important. A scheme which makes these operations particularly easy, is described in [6] and sketched in the following.

The setup of this accumulator scheme uses an RSA modulus n, a seed v and the IDs of the users denoted as ID_i. The accumulator value is built with

$$z_v = v^{\prod_i ID_i} \pmod{n}.$$

The values z_v and n are published and a witness value $x = v^{\prod_{i \setminus j} ID_i} \pmod{n}$ is generated for each value ID_j contained in the accumulator to confirm it is contained in z_v. The latter can be achieved as the verifier is able to compute $x^{ID_j} \pmod{n}$ which is per definition equal to the accumulator value z_v.

A limitation of this scheme is that upon the addition of a value to the accumulator all witnesses need to be recomputed. Camenisch and Lysyanskaya have presented a solution in [6] which they called *dynamic accumulator*. There are several schemes proposed to attain a dynamic accumulator but all come with constant computing time, i.e. it is not depending on the number of values already in the accumulator.

[3]Public Key Infrastructure

A first possibility is to change z_v upon a new user with ID_{new} to $z'_v = z_v^{ID_{new}} \pmod n$ and the witnesses have to be changed accordingly with $x' = x^{ID_{new}} \pmod n$. Revocation is more elaborate because not only that the revoked ID ID_{rev} needs to be extracted from all witnesses but also the target value, i.e. z_v is adapted. The accumulator is straightforward changed to $z'_v = z_v^{1/ID_{rev}} \pmod n$ and the witnesses are determined by $x' = x^b z'^a_v \pmod n$ such that $a \cdot ID_j + b \cdot ID_{rev} = 1$. This follows through the following calculation where in the last step the condition from above is used.

$$\begin{aligned} x'^{ID_j} &= \left((x^b z'^a_v)^{ID_j}\right)^{ID_{rev} \cdot 1/ID_{rev}} \\ &= \left(z^b_v z'^{aID_j}_v\right)^{ID_{rev} \cdot 1/ID_{rev}} \\ &= \left(z^{bID_{rev}}_v z^{aID_j}_v\right)^{1/ID_{rev}} \\ &= z_v^{1/ID_{rev}} \end{aligned}$$

A second possibility relies on the issuer knowing the factorisation of n. The scheme can be further improved, resulting in not changing the accumulator when a new member joins and calculating new witnesses only upon revocation. Refer to [6] for more details.

Chapter 4

Discussion of Selected Authentication Systems

As more and more services become accessible from remote locations, authentication gains in importance. This results, as accessing a service over a network does not allow the use of standard physical tokens, such as a passport or a driver's licence.

Today, the usual way of performing authentication is by using a username-password system. However, there are settings which require more sophisticated approaches. A PKI-based authentication system is one of them. Unfortunately, it suffers from several disadvantages, too. The following sections provide an overview over security-relevant aspects of both systems as well as an introduction to credential systems. These systems provide the users with far-reaching privacy. As they still are being designed there is much space left for optimisation and for extending the system. Before we present one such optimisation, we will discuss the three above mentioned systems.

4.1 Username-Password System

Today the most-commonly used authentication scheme is a *username-password* approach. In such a system, a user chooses a username and a corresponding password when he first wants to authenticate to a service providing organisation. We refer to the username-password combination also as authentication tuple. In addition to the authentication tuple, the user is required to provide additional data. This could be the name, address or date of birth of the user. Successive visits are authorised if a valid authentication tuple is provided. The advantages of this system are particularly the ease of implementation and use. Notably, there is no user-side deployment required.

It has severe disadvantages when regarding the security. The biggest problems are not due to system design but to typical user behaviour. In particular, users who have many accounts tend to choose the same authentication tuple. Thereby, colluding organisations can link user actions and security issues at one organisation are a risk for the authentication to another organisation, too. Another security issue arises due to most users not managing their different authentication tuples appropriately. For example, they store all used username-password combinations unencrypted in a file on their computer.

All the same, there are also weaknesses due to the design. An important one is that user actions carried out with one username at one organisation are linkable. This fact decreases the privacy a user can benefit from. Another weakness, which is privacy relevant, is caused by the additional data which the user is supposed to provide. Mostly this data is acquired due to legal obligations or for marketing purposes. However, if not properly checked, the organisation cannot rely on the given information as the user might provide wrong data on purpose. The consequence is a reduction of the privacy of honest users whereas dishonest users contribute to the dissatisfaction of the organisations.

A disadvantage having special importance, regarding the goal of this thesis, is the possibility of users to share their authentication tuple. Especially, there is no binding of a username to a real person. The linkability of the system still offers organisations to run probabilistic algorithms to detect massive sharing. Then again, there is no possibility to detect duplicate use of the authentication tuple itself. As an example, we will look at two users sharing a username. If both users make use of the account only seldom, they might still authenticate fewer times compared to one user making heavy use of the system. Consequently, the non-shared authentication is rather suspected to be shared than the shared authentication tuple. On the other hand, a username shared over the Internet is suddenly used by hundreds of users and the organisation will recognise the increase in authentications.

In Fig. 4.1 we can see that the security of the system is heavily relying on the choice of the password. For example, especially the username but also the password are vulnerable to a guessing attack as the username is mostly in direct relation to the name and surname of a user. This attack is indicated by the "guess" box at the lower right corner of the figure. Another example might be the use of security flaws in some software running on the target host which makes it easy to extract data from this host. This attack is depicted as "3$^{\text{rd}}$ party weakness". The attack vector indicated at the left side of the figure refers to the possibility of extracting the data not from the user itself but to retrieve it from the organisation which the user authenticates to. At the organisation the attack vectors are equal to the possibilities at the user's side although the probability of success is very different. Unfortunately, many service-providing organisations only give advice on the choice of a strong password. They do not enforce compliance with guidelines. The latter regulate for example the length and the number of different character types that need to occur in a password. This measure limits the success of password guessing and brute force password extraction attacks.

As already mentioned, nowadays a user has affiliations with several service providing organisations. To reduce the entropy he has to store, the user tends to choose the same username-password combination for several systems. The security threat lies in the possibility to gain information for several systems. Even worse, the accounts accessible with the same authentication tuple have possibly different value and are therefore protected differently by the respective organisation. For example, a bank protects the eligible access combinations much better compared to a social networking service. Exemplary the attack on the social networking system could allow for an attack on the bank account. This attack is not depicted in Fig. 4.1, although it would only mean to add another organisation where all the attack vectors again apply. The probability of success of such an attack can be calculated by multiplying the probability that the user uses the same authentication tuple with the probability of a successful attack.

Currently, settings with high security requirements, e.g. in a banking environment, extend their system by a token. The user not only has his username and password but also a token. For the authentication, the organisation sends a challenge to the user. The token is able to computes a response from this challenge. Successful authentication requires the triple built of username, password and correct response to the challenge. Thereby, a substantial increase of the overall security can be achieved at the price of less convenience for the user. He only accepts such an inconvenient system for high value accounts where the need for increased security is obvious.

Looking at the attack tree for the basic username-password system depicted in Fig. 4.1 the attack possibilities are massively reduced with the addition of such a token. In particular, the brute force and password guessing are much less promising. In addition, stealing the database entries is not sufficient as the correct response of the token is required for authentication. We are not evaluating this scheme further as inherent problems, especially the lack of privacy and manageability for the user, persist.

4.2 PKI-based System

A very different security is offered by a *PKI-based* authentication scheme. The user gets an asymmetric key pair, which he will have to use for authentication with an organisation. The increase

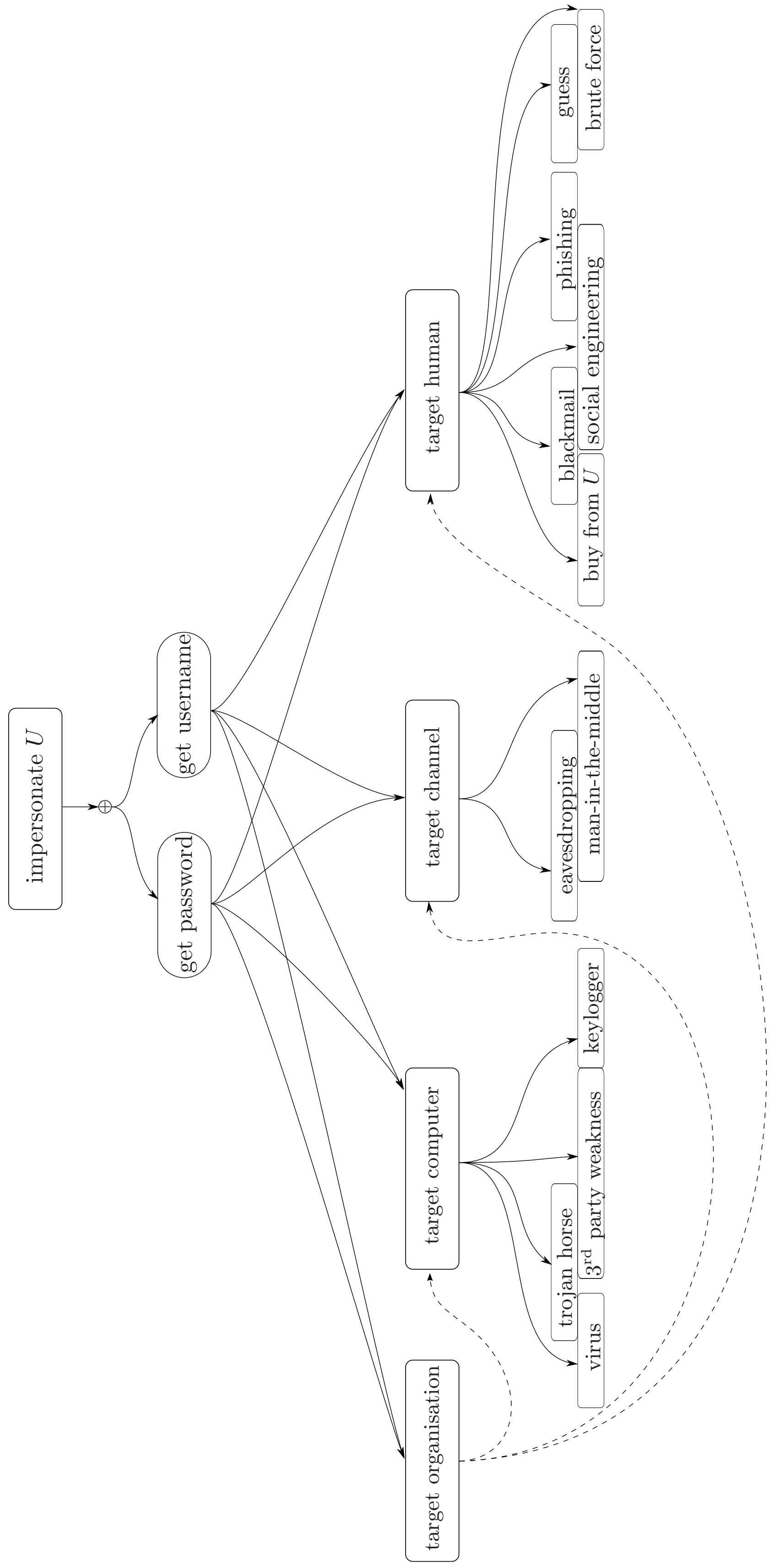

Figure 4.1. Attack tree for a username-password system

in security results from the reduction of attack vectors. In particular brute force and guessing attacks as well as the use of a keylogger and social engineering are not promising anymore. Clearly, the verifier still is able to link all transactions. This results as the user authenticates with respect to his public key.

Regarding the aspect of sharing prevention, the advantage of this system is that a software token, i.e. the key pair, is needed in order to authenticate. It cannot be transferred as easily as a username-password combination as most users are not capable to store it in their brain. However, the transfer from one computing device to another is as simple as in the username-password system. Hence, no effective sharing prevention is attained.

PKI-based systems are currently not widely used. The reason for the small number of implementations might be the difficult deployment on the user's side. In many use cases, the user cannot be expected to master the difficult setup. In addition to that, the management of a large number of keys would cause even more trouble than the management of a large number of username-password combinations. Subsequently, the drawbacks of a PKI-based setting are only tolerated with high-value credentials. Particularly, such credentials are much less subject to sharing as the user is not willing to do so. Duplicated credentials still need to be regarded as for example malware could extract the authentication information.

4.3 Credential Systems

Credential systems are currently not widely used in practice. However, they offer a wide range of opportunities wherefore those systems are of enormous interest in the research community. A number of properies make them superior to username-password systems. Due to this reason we provide an extension to a credential system. We want to analyse the system in more detail compared to the systems described so far.

A credential system has first been introduced by Chaum [11] and it has been extended various times. This thesis relies mostly on the scheme as described by Camenisch et al. in [5]. A brief introduction into the basics of credential systems and in particular into anonymous credential systems is provided. To be able to explain such a system, some terms need to be defined. The definitions provided here are not rigorous but serve more to clarify the reader's intuition.

Definition 7 (Pseudonym)**.** A Pseudonym is an identifier used to associate a user with his resources. Even so, the pseudonym is used to bind credentials to a specific user. Throughout the paper, a pseudonym is abbreviated as "nym".

Definition 8 (User)**.** A party using the given system and being defined by having one unique master key pair consisting of a master secret key (MSK) and a master public key (MPK). A user can get several nyms which all are bound to her MSK.

Definition 9 (Organisation)**.** An entity in the credential system having special properties such as the ability to issue credentials or to offer some other service. Service-providing organisations mostly act as verifier of either credentials, nyms or other attributes.

Definition 10 (Trusted party)**.** An entity being able of performing any operation. The results of its computations are trusted by all other entities. Also it is trusted not to share knowledge of the values it has been given.

Definition 11 (Semi-trusted entity)**.** Similar to the trusted party with the difference that the trust relation only holds for a specified entity.

In a large scale, highly-connected system, the privacy of the users becomes an issue. This issue can be addressed using a pseudonym system. In such a scheme each organisation does only know its users by pseudonyms. Depending on the type of service that the organisation offers, each user might have even several nyms with one organisation.

Example 4.3.1. It is not possible to have more than one pseudonym with the government. However, a user might have several pseudonyms with one bank. This corresponds to the real world situation where one person might have several bank accounts but only one passport.

The introduction of a pseudonym system has severe implications for all organisations. They are not necessarily able to link the pseudonym to a real person. Clearly, the privacy of users cannot come at the cost of insecurity on the organisation's side. A suitable system, valuing both sides of a commercial relationship, could speed up the growth of e-commerce considerably. A simple pseudonym system, as the one just described, shows already some privacy benefits. A more complete system with several other properties is used in the following. This system is called anonymous credential system or *idemix* which is the name of an implementation of an anonymous credential system by the IBM Zurich Research Laboratory.

4.3.1 Basic Setup of a Credential System

To set up a credential system three roles are required: a user U, an organisation O_I issuing certificates[1] and an organisation O_V verifying certificates. Certain statements apply for any organisation. Instead of specifying that the statement applies for both O_I and O_V, we denote the organisation that case as O. In some cases it is necessary to have a trusted party T or a semi-trusted party T_s but the reliance on such parties is reduced as far as possible as trusted parties are in practice hard to establish.

Notably, a credential system is a means to enforce privacy and anonymity on the application layer and therefore it provides no possibility for protecting the user on a lower layer, for example the network layer. Consequently, the anonymity provided by the application only applies if the communication channels support anonymity as well. For example, onion routing schemes such as TOR[2] or JAP[3] or mix networks support anonymity on the network layer.

Pseudonym generation

Generating a credential is based on the organisation knowing the user U either by the real identity or by a pseudonym. The first situation is necessary for credentials, such as a passport or a driver's licence, the second is the general case. Regarding the privacy, the first case should be limited as far as possible. Nonetheless, a credential obtained in an identifying session can be used to acquire further credentials in a "pseudonymised" session. Thereby, the organisations issuing the pseudonymised credentials still benefit from a good security if they trust the issuer of the identifying credential.

A nym $N_{U,O}$ is established before a credential can be generated. This action is necessary independent of the knowledge an organisation has about the user U. It is a fundamental property of the system that O cannot link any two transactions of U through the use of the credential. In particular, even if a credential C received under a pseudonym $N_{U,O}$ is used several times to authenticate at organisation O. Similarly, the usages of same credential C utilised to authenticate to another organisation O_i cannot be linked. The notation of the nym is chosen such that it is visible which user U and organisation O know the nym. Still it does by no means imply that the organisation is aware of any information about the user.

An important fact is that nyms are bound to the MSK of a user. This is achieved by cryptographic means. For example, a Pedersen commitment on the MSK can be used as nym.

Credential generation

As soon as a pseudonym has been established, the issuing organisation can issue a credential to a user known under nym $N_{U,O}$. The organisations issuing credentials are usually called identity providers as they provide the user with signed attributes. Those attributes can be used within the system by U, i.e., they can be proven in zero knowledge to any verifying organisation.

In a practical credential there need to be user-chosen, jointly-established and organisation-chosen attributes which can be certified in a credential. The user-chosen ones are provided by the user

[1]Certificates and credentials are treated as synonyms throughout the paper.
[2]further details can be found on http://www.torproject.org/
[3]further details on http://anon.inf.tu-dresden.de/index.html

similar to filling in a form in current systems. Jointly-chosen attributes are used for jointly-chosen random values which relieves the communication partners of the establishment of a trust relationship. The last type of attribute, organisation-chosen attributes, is used for the attributes that are certified by the organisation.

The credential is obtained by U upon a successful execution of the issue protocol. It needs to be stored for further use. However, it is never to be handed over in an unaltered manner to another party. This also applies if the credential is shown to a verifier.

Showing Credentials

Using a credential to authenticate to a verifying organisation O_V involves showing certain attributes as well as proving legitimate possession of the credential. The second requirement can be met by showing knowledge of the MSK that has been used for the nym establishment. All the same, this proof is not satisfactory as U might have shared his credential including the MSK. There is no other assurance for the verifier that this has not happened except if the sharing of the MSK is aggravated by some means. The proof of knowledge of the MSK is accepted at this state.

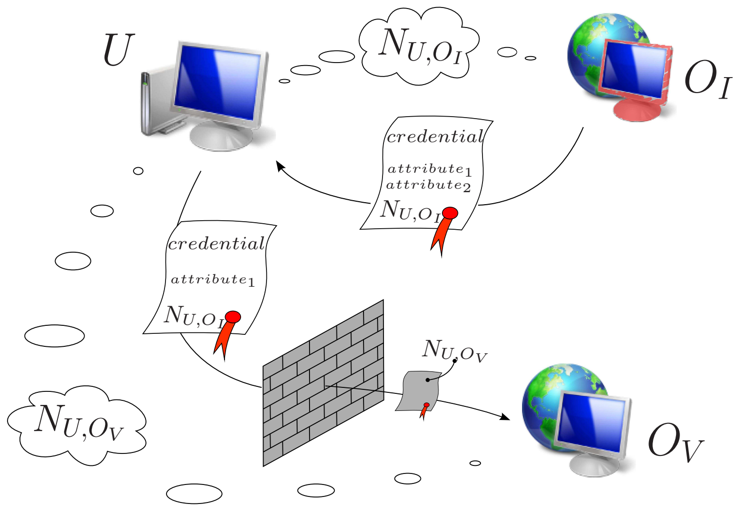

Figure 4.2. Credential generation and showing to an verifier: O_I issues to U a certificate under nym N_1 upon the attributes $attribute_1$ and $attribute_2$. U wants only $attribute_1$ to show to the verifier O_V which is done under nym N_{U,O_V}.

Some parts of the lifecycle of a credential including the acquisition and the use are depicted in Fig. 4.2. The zero-knowledge proof is illustrated as a wall. When U wants to use the credential, he needs to randomise the credential and issue a zero-knowledge proof of knowledge of the randomised credential. He can choose which attributes are to be revealed. However, the fact that the pseudonym N_{U,O_V} is related to the MSK needs to be revealed as well as the signature from O_I. Still, the nym used with O_V does not need to be the same as the one use with O_I. Nonetheless, they necessarily need to be issued on the same MSK. The signature of O_I is also hidden using a zero-knowledge proof. Hence, the verifier does not learn anything except for the fact that the attributes shown in the certificate have been signed and handed over to the person known under nym N_{U,O_V}. This allows U to access the service or resource pseudonymously and he is not linkable through the use of the certificate. U needs to have no personal data at the organisation, e.g. with a newspaper subscription, he does not need to reveal any information except the valid subscription. The user can subsequently read the newspaper anonymously. Both cases are practical but the anonymous authentication is limited to services not being associated with a personal account at the organisation.

In Fig. 4.2 the credential is shown under a different pseudonym than it has been issued to. This procedure is called *credential transfer*. The name might imply that a credential is transferred from one user to another. In fact, the transfer can only happen from one pseudonym to another if both pseudonyms are built w.r.t. the same MSK. The MSK is unique for each user, wherefore the transfer cannot happen from one user to another. The binding of the MSK to a user is discussed in detail later on. This is necessary as credential misuse can, for example be achieved by sharing the MSK.

4.3.2 Properties of Credential Systems

There are two main objectives of a credential system, which are the protection of the privacy of the users on one hand and the security of the resources provided by organisations on the other hand. In traditional systems those properties are conflicting goals. To depict this conflict, we have a look at two systems. Either of them is optimised for one of the conflicting goals. The first system achieves full anonymity for the users whereby optimal privacy can be attained. The second system could implement strong authentication combined with identification of the users and hence achieve a high level of security for the service providing organisations.

The currently used anonymous credential system has a number of properties which protect the users and the organisations. The most important among them, serving the protection of the organisations, is that pooling of credentials is impossible. For example, a user can not use credentials that he has obtained in combination with credentials delivered to another user. In particular, a credential can only be shown when the MSK of the nym is known. Combined use of credentials is only possible when the used nyms base on the same MSK. In other words, the MSK upon which the nyms of the credentials that are used together must match. As mentioned earlier, they must be built using the MSK. As a consequence, different users cannot pool their credentials.

The privacy offered by an anonymous credential system actually goes as far as not even colluding organisations can reveal identifying information about their users. This property could be undermined by requiring the user to reveal his name. However, the system design allows the user to reveal only the attributes he is willing to show. Thereby, no identifying information is given to the organisation without the user getting to know.

The advantages of an anonymous credential system over the username-password system can be visualised by a comparison of their respective attack trees. The attack tree of the *idemix* credential system is given in Fig. 4.3. Figure 4.1 depicts an attack tree for a current username-password system. An obvious discrepancy is the lack of attack vectors targeting either the channel or the verifying organisation. This results from credential systems using zero-knowledge proofs for authentication and hence not leaking information that can be used in a successive authentication. In particular, there is no information about the actual credential or the MSK leaked to the verifying organisation. In addition to those advantages, it also gets harder to convince the user to give the credential information away. This is due to this operation not being a standard operation because a credential is never to be given away in an unaltered manner. The anonymous credential system, however, needs to offer an operation allowing the user to copy a credential from one device to another. Still, this operation could be associated with many agreements that the user needs to allow, which make a social engineering attack much less likely to succeed. An important finding of this comparison is that it gets much harder to acquire the MSK form the user. Especially, the MSK is needed within the credential system only, whereby acquiring this critical information is no longer possible without the user's knowledge or due to implementation flaws.

Single-show vs. Multiple-show Credentials

Single-show credentials can only be used one time. They can, for example, be used as electronic cash or as vouchers. If a single-show credential is given away deliberately, this action translates to a gift in the real world, meaning that the donor cannot use it anymore. The only problem which might arise is that the presentee is legally not allowed to use the credential. Yet, this problem is not in the scope of the thesis.

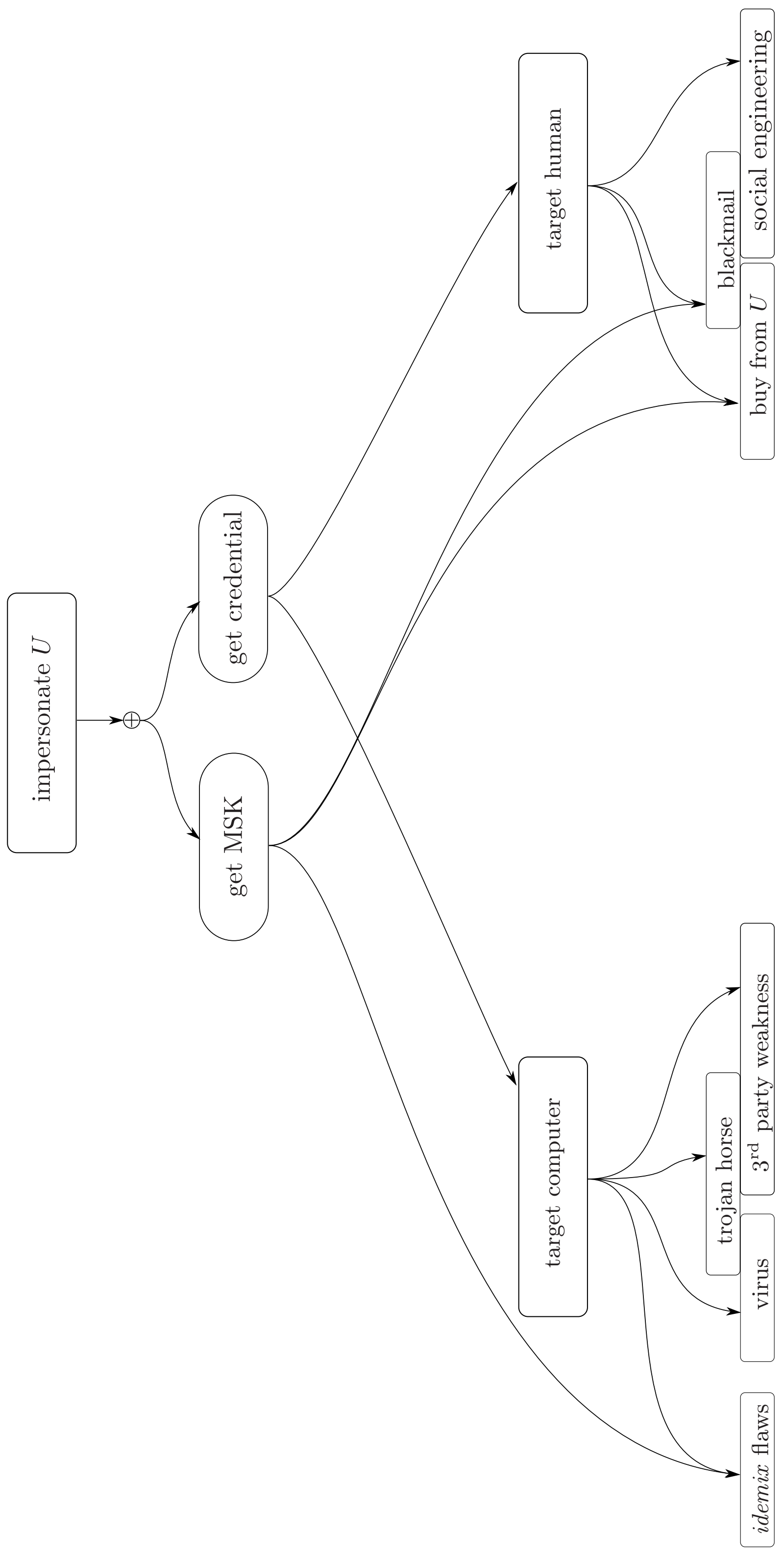

Figure 4.3. Attack tree of the *idemix* anonymous credential system

For such credentials only double spending is a serious problem. A solution to this problem has been proposed by Camenisch et al. in [4]. The intuition is to use a serial number and a double-spending number which encodes the pseudonym of the user in each credential. Multiple use of a credential can be detected through the serial number and the double-spending number allows revealing the pseudonym. The extraction of the pseudonym allows for punishment of the respective user. We do not provide a detailed description to this solution as the intuition of the scheme is the same as the one used for the K-show credentials. K-show credentials are discussed in extensive detail in Section 6.1.

Multiple-use credentials, on the other hand, are to be handled very differently as they allow sharing. They could basically be used by several users at the same time even though only one user had the right to use the credential. Organisations issuing such credentials demand protection against such misbehaviour of users. Dealing with this problem is the main objective of this thesis.

Credential Revocation

The challenge of revoking a credential is similar to revocation of public keys in a PKI. Digitally signed objects have to be made invalid although they verify correctly. One solution is to publish a list of revoked credentials which does not scale as the length of such a list is linear in the number of revoked credentials. Additionally this solution is not privacy aware.

As a first proper solution, embedding an identification number ID_U of user U into the credential is proposed. The credential is consequently of the following form: $d = \ldots \cdot \ a^{ID_U} \pmod n$. Additionally a list of all valid identification numbers is to be published. U then proves that his ID is on the list by computing $G_i = g^{ID_U}$ and then proving the following:

$$PK\{(\ldots,\gamma) : d = \ldots \cdot a^{\gamma} \pmod n \wedge ((1/G_1)^{\gamma} = 1 \vee \ldots \vee (1/G_k)^{\gamma} = 1)\}$$

Alternatively, a list with all invalid IDs could be published and the user has to prove that his ID is not on the list. The impact of the choice whether to publish the valid or invalid user IDs on the security of the system is similar to an access policy which is "accept" or "reject", respectively. The main disadvantage of this solution is its linearity in the number of either valid or invalid user IDs which makes it inefficient for large groups of users.

A second approach is using the inclusion of the user ID in the credential as seen above and a list of all valid IDs is published as $\{ID_1, \ldots, ID_L\}$. The user chooses a random h and computes $h_u = h^{ID_U}$. He also proves to the verifier that the ID in the credential and the one used for the computation of h_u match, i.e.

$$PK\{(\ldots,\gamma) : d = \ldots \cdot a^{\gamma} \pmod n \wedge h_u = h^{\gamma}\}$$

The verifier can make sure that there is an $i \in \{1, \ldots, L\}$ for which $h_u = h^{ID_i}$ where ID_i needs to be on the list. To reduce the complexity, the verifier could precompute the list resulting in a lookup table. However this leads to linkable actions of the user as h cannot be chosen arbitrarily by the user anymore. This could be avoided using a hash function $h = \mathcal{H}(verifier, time)$ where the verifier could precompute the lookup table and h would change with every use. [5]

A more sophisticated approach is the use of *cryptographic accumulators*. An accumulator can contain many IDs which are all associated with a witness. It is not possible to compute a witness for a value which is not included in the accumulator. The witness needs to be presented to prove that a given ID is contained in the accumulator. The accumulator hence substitutes the list of valid users, which is sensible as it has some favourable properties. Especially, the accumulator is much shorter compared to a list of IDs. Also the scheme proposed by Camenisch et al. in [6] allows the implementation of dynamic accumulators. They enable efficient update of the accumulator upon the addition of a value. Even better, one scheme offers accumulators which need only to be changed upon removal of a value. A more detailed description can be found in Section 3.4 on page 24.

Anonymity Revocation

The term “revocation” is needed in two entirely different situations. Firstly, a user recognising that his credential has been stolen wants to revoke this credential. Credential revocation is also needed if the organisation suspects the user of the illegitimate use of his credential. This type of revocation has already been described above. Secondly, if an organisation knows that a user of a credential performs some sort of illegal action, it may demand to reveal the identity, i.e., revoke the anonymity.

Anonymity revocation can either be executed in a local or a global manner. Local anonymity revocation means the creation of the link from an authenticated user to the nym he is using with the issuer of the credential. Upon a defined action, the organisation is able to reveal the pseudonym of the user. It can subsequently revoke the credential. This can only be done when the user agrees to the procedure beforehand. Even so the actions which allow for revocation of the anonymity have to be defined upon issuance of a credential. Moreover, a user-side semi-trusted party[4] is needed to execute the revocation after the organisation proves that a violation has occurred. Global revocation reveals the identity of the user upon having performed illegal actions. The necessity of this option is given as illegal actions, even though they should be limited through the system design, must be punished appropriately.

[4]refer to page 30 for details

Chapter 5

Analysis of Misuse Protection Mechanisms for Anonymous Credentials

As already mentioned, we are extending anonymous credential systems with misuse protection mechanisms. The choice of appropriate mechanisms is crucial to the acceptance of the system. To provide a well-founded choice, the most promising approaches for misuse protection are explained in the following. Their respective advantages and drawbacks are listed in order to simplify the choice of the mechanism to be implemented. The description of the techniques is kept at a high level. The cryptographic details do not help a better understanding of the properties of the systems. The interested reader is referred to the research papers, in particular to [5], [3], [2] and [1].

When designing a system, the focus needs to be on the possible advantages such a system can offer for the participants. Especially, its advantages ov er the system it should substitute need to be taken into account. The organisations in today's systems require complete identification of their users and even urge them to provide more data than necessary. Considering this situation, the organisations have no reason to change to a new system, especially if the risk of systematic misuse is higher compared to the system in place. When looking at the current system more closely, we find that many organisations do not really look for identification. The identifying information is a means of gathering marketing information or simply to comply with legal obligations. As we have already seen, the security provided by such information is negligible. The lack of a trust relationship between users and organisations suggests the introduction of a strong authentication mechanism. We propose this mechanism to be an anonymous credential system, e.g. *idemix*.

Currently, anonymous credential systems lack an effective misuse protection mechanism. Before going into details of misuse protection, we need to elaborate on different cases of misuse. We believe that credential misuse is predominantly associated with the use of duplicate credentials. Thus, misuse protection is eventually the prevention of duplicated credentials. The vision of achieving not only the prevention from duplicates, but also a binding to one specific user is targeted by biometrical systems. The combination of such a system with the system developed here is believed to be straightforward. We consequently focus on the protection from duplicate credentials. Hence, achieving protection against shared as well as stolen credentials at the same time.

One can argue that a stolen credential could still not be prevented as long as the theft is not recognised by the legitimate owner. This argument certainly holds but it translates directly to the theft of any physical object: if not detected, the thief can use the object at his discretion. As an example, a stolen credit card can be used as long as the theft is not reported and the validity of the card is not revoked. Hence, the challenge is not to prevent the theft itself but achieve that the victim will be aware of the theft. Trying to prevent theft itself will be fruitless as there always is the possibility to threaten a user to give away his credentials.

There are different ways to attain misuse protection[1] in an anonymous credential system. The requirements of different approaches are very different as are their implications on credential misuse. We will structure the approaches in software-based and hardware-based ones.

5.1 Software-based Misuse Protection

In this section the protection mechanisms based on cryptography are analysed in more detail. There is a broad range of possibilities to address the different misuse scenarios using cryptography. Given the broad range of possibilities to counteract the problem there is also a broad range of approaches that have been studied.

Despite there merits, there is one particular thing that cannot be achieved with sole cryptographic means. This is a limitation of the number of copies that can be made from an original credential. As it is digital information, there exists no possibility to inhibit the copying procedure. The approaches given in the following are still useful as they provide other means which inhibit users from using copied credentials. The ideas behind the mechanism is not to inhibit the copying but to either make users unwilling of credential misuse or to design the system such that the issuer maintains tight control of the credentials.

5.1.1 All-or-nothing Non-transferability

The basic idea of all-or-nothing non-transferability is to tie the credentials of one user together. Assuming that a user owns at least one valuable credential, he will not copy even one of his credentials. If he would do so, the recipient of the copy was able to use all the credentials of the misbehaving user.

The method to achieve the binding of the credentials is the publication of the encrypted credential as soon as it is being established. The published credential is encrypted using the secret key of the user. If a user U_1 wants to share one of his credentials with U_2, U_1 needs to reveal his secret key to U_2 as otherwise the shared credential cannot be used. With the secret key, however, the recipient of the copied credential would be able to use all the other credentials U_1 owns. This follows from the fact that copies of all credentials are publicly available. Furthermore, the decryption of the published credentials of U_1 for U_2 is possible as he knows the secret key of U_1. [5]

This approach is powerful if users have at least one valuable credential in the anonymous credential system. However, it does not offer a generic protection against duplicated credentials. The protection is entirely based on the willingness of the user not to share one credential and therefore not to share any of them. The issuance of credentials could be inhibited if an organisation is not convinced that the sum of a user's credentials is of enough monetary value to make him unwilling to share. In this case, the organisation might suspect that the user will be willing to share all of his credentials and subsequently not proceed with the issuance.

Another issue is the lack of protection from theft. If the secret key is stolen, the thief can easily retrieve all the credentials and make use of them. There is a positive side of the credentials being publicly available, too. A user not having a local copy of the credential can retrieve the missing credential. This fact could be exploited if the secret key was stored on a memory-constrained device. The device would not have to store the credentials locally and still it would have the possibility to retrieve a credential as soon as it is needed.

5.1.2 PKI-assured Non-transferability

PKI-assured non-transferability is similar to the all-or-nothing approach as the mechanism bases on the user's willingness to achieve its goal. Still, instead of binding the credentials to each other, the credentials are tied to a valuable secret outside of the system. For example, a credential could be secured with the credit card information or the bank account. The system allows each

[1] *Misuse protection* refers to the prevention of duplicate credentials in this thesis.

credential to be bound to a different valuable outside secret. This is an advantage over the all-or-nothing approach. The case where the organisation is not willing to issue a credential can be avoided. A challenge, though, is to find an appropriate value to bind a credential to. The organisation and the user have conflicting goals when choosing an outside value. The user would like the outside value to be as low as possible. This results as theft of the credential necessarily implies that the outside value is stolen as well. The organisation on the other hand wants the value to be as high as possible, as otherwise the user might still share or sell the credential.

As already mentioned, the all-or-nothing and the PKI-based approach are helpful but, in addition to other deficits, they rely on the cooperation of the user. For the PKI-based approach, the outside value needs in fact to be higher than the value that could be achieved by selling the credential. The latter is difficult to estimate. Moreover, a theft of credentials results in the loss of the outside value of the user. It would be difficult to justify this additional attack vector against the outside value.

As the all-or-nothing non-transferability, this scheme adds no security protection against duplicate credentials. They can neither be detected nor prevented. General misuse is subsequently not prevented by either scheme. Malicious software and imprudent users cannot be protected with such an approach. For those resons, we do not consider this approaches to lead to an effective misuse protection in an anonymous credential system.

5.1.3 K-show Credentials

The scheme proposed by Camenisch et al. in [3] approaches the problem from a different angle. A mechanism for detecting credential overuse is provided by so called K-show credentials. A probabilistic reduction of the number of instances can be achieved. The credentials are slightly extended such that they can only be shown at most K times during one epoch. Especially, there is a cryptographic mechanism which reveals certain information if a user overuses a credential. In particular, the pseudonym information between the user and the issuer can be extracted if the credential is shown more than the allowed number of times. The definition of the epoch and the number K can be chosen for each credential individually.

A drawback of the scheme is the lack of a mechanism to minimise intentional sharing. Depending on the choice of the parameters, sharing with a few people cannot be detected. In other words, there is no means of detecting a difference between a few instances which are rarely used and one instance which is very frequently used. However, with the appropriate choice of the parameters, the limit on the number of instances that can exist undetected can be kept very low.

As opposed to the schemes given previously, this setting can counter general credential misuse. The limitation on the number of usages of a credential might not look as a big advantage at first, but it has far-reaching consequences which will be discussed in more detail in chapter 6.

5.1.4 Counter-based Credentials

In the setting of counter-based credentials, the credentials are issued with a counter and a signature on the counter. When a credential is used, i.e., it is shown to a verifier, the counter needs to be incremented. Additionally, the prover has to show the signature on the counter that it wants to use. Certainly, a successful authentication with a verifier causes the latter to issue a signature on the incremented counter value. The details of the approach are to be published by Heydt-Benjamin et al. [16].

The most unfortunate property of the scheme is the need for the issuer to be online. This is necessary for the signature on the incremented counter. The efficiency of the scheme, though, is promising. As mentioned several times before, the duplication of credentials cannot be inhibited by a cryptographic approach. All the same, counter-based credentials generate high cost in communication if duplicated credentials want to be used. Those costs arise due to the necessary synchronisation. A way to undermine the duplication protection is to run an online entity which manages the counter and its signature. Yet, this online entity is costly as well.

Similarly to a counter, the entity could issue single-show credentials after each successful credential show. Such a scheme is computationally intensive. On the other hand, it enables the use of protection mechanisms such as the double spending test. A costly attack would again be an online entity administrating the authentications of all sharing entities. [5]

5.2 Hardware-based Misuse Protection

The general idea of hardware-based credentials is to bind a credential to a tamper-resistant device. In a credential system, the binding can be achieved by hiding the MSK[2] of a user in the device.

The credential can subsequently not be transferred to another device anymore without the extraction of the MSK from the tamper-resistant device. The problem arising due to credentials being digital information can hereby be effectively countered. In other words, the particular difference to the previously described methods is that it is much harder to duplicate data stored on secure hardware compared to duplicating data stored on a standard device, i.e. a personal computer.

A big advantage of this approach lies in the fact that a highly tamper-resistant piece of hardware is included in the process of using credentials. The existence of duplicated credentials consequently implies breaking the tamper resistance. Although being possible, it is associated with very high cost.

The resource constrains of a tamper-resistant device, however, cause difficulties regarding privacy. The security device is usually not capable to execute all computations necessary to proof possession of a credential. Subsequently, it makes use of a host which is supposed to execute the non-security-critical operations. The tamper-resistant device, on the other hand, will have the task of computing the security-critical values. As all communication of the security device is managed by its host, privacy assurances cannot be given. For example, the host could simply append an identifying value to each message. Consequently, the host can negate the privacy even in the presence of a tamper-resistant device. Therefore, the security device is not expected to compute privacy respecting values.

5.2.1 Challenges of Binding Credentials to Hardware

The methods used to bind a credential to a tamper resistant device differ depending on the kind of hardware being used. The basis for binding a credential to a TPM[3], for example, is described by Camenisch in [2]. The binding to already specified hardware is difficult but in the case of the TPM it is possible. The binding to other hardware, e.g., smart cards, is less complex as the device would most probably be designed after the specification of the necessary operations.

Authentication to the hardware device is an important part in the overall system security. As an example, sharing would be possible even if the credential was reliably bound to a secure device if the device itself can be passed to another user easily. Hence, the binding of the hardware to the user needs to be studied as carefully as the binding of credentials to the secure device. A possible solution is provided in [22] but coverage in detail is out of the scope of this paper.

5.2.2 Embedded Security Device

There is a tendency to embed security devices into standard hardware. The Trusted Computing Group[4], for example, promotes trusted building blocks. Among those building blocks is the tamper-resistant chip called Trusted Platform Module (TPM). Having the largest market penetration, we study the possibilities of embedded secure hardware on the example of the TPM.

[2]Master Secret Key

[3]Trusted Platform Module specified by the Trusted Computing Group (TCG)

[4]The Trusted Computing Group is an industry conglomerate investing into the development, definition and promotion of open "standards for hardware-enabled trusted computing and security technologies"[23].

Advantages

The advantage of the TPM is not only the fact that it is already embedded into nowadays computers. The use of an embedded security device relieves the user of any action except for the authentication to the security device. Even better, when the user uses his credentials frequently, authentication to the TPM is only needed the first time. Subsequent usages, occurring within a defined time interval, can be carried out without re-authenticating. The system security certainly is reduced by this measure but it is tolerable compared to the usability which is gained. The measure should simply show the excellent usability of an embedded security device. Essentially, the protocol realises a credential system with additional device attestation semantics to make assertions about the state of the device.

Another advantage is the availability of methods to bind credentials to a TPM. The method described by Camenisch in [2] extends the direct anonymous attestation (DAA) protocol. DAA, as specified in [1], allows a TPM to prove that it is genuine while protecting the privacy. The DAA protocol defines an attester, a host, a TPM embedded in the host and a verifier to be present. The TPM communicates only through the host it resides in. The primary fonction of the TPM is to store the secret (m_0, m_1) which is split into two values for efficiency reasons.

When the host wants to prove that it embeds a genuine TPM, it first asks the TPM to generate a commitment on the secret (m_0, m_1). The TPM additionally provides a proof that it has gotten a signature from an attester on the commitment. Commitment and proof are sent to the host. As already mentioned, the information sent to the host is not anonymised. As the TPM is only communicating through the host with any external device, privacy enforcements by the TPM could be subverted by its host. For example, the simple addition of a unique value to each message would allow identification and linkablity by the organisation which undermines the privacy of the user.

Assuming to deal with an honest host, the latter anonymises the data received from the TPM and sends it to the verifier. The protocol is computed mostly on the host. Only security-relevant operations, i.e. operations which need knowledge of either m_0 or m_1 are performed on the TPM. Therefore, the DAA protocol can be extended to include other attributes than m_0 and m_1. The host would add the attributes similar to the secrets residing in the TPM. For details on the extension we refer to [2].

Disadvantages

The biggest disadvantage of using an embedded security device is the fact that there is no bijective mapping from users to devices. For example, a user might own several devices. This causes problems as the users must be allowed to transfer their credentials from a device to another. Furthermore, there must be the possibility for several users to use one security device. In the current setting this would imply several MSKs to be stored on one security device.

To solve the problem of a user having several computing devices, a transfer of credentials could be envisioned. Another possibility to overcome the limitation is a separate credential for each of his devices as described in [2].

As complicated systems tend not to be accepted by users, we studied the transfer of credentials from one TPM to another. The general idea is to issue a credential to a dynamic accumulator instead of directly binding it to the TPM. Another binding between the accumulator and the TPM is established thereafter. The TPM receiving the credential would be able to reissue credentials to another TPM. It will be called the *master TPM*. The reissued credential would be bound to the receiving TPM. The master TPM is to be the only TPM able of reissuing credentials. Two fundamental problems could be solved by this approach. First, users having several devices could easily use them by having issued the credential to one TPM and re-issue them to their other devices. Second, the migration of an entire system of a user would be largely simplified. However, the second scenario calls for another method which allows to transfer the master TPM role to another TPM.

This directly leads to the problem of revocation. An efficient solution is necessary as a typical user might change his devices frequently, which leads to masses of revoked credentials. There must be a possibility to revoke a credential issued to a TPM without revoking the attestation of the TPM. For example, when the user sells his computing device. Nevertheless, the revocation of a TPM must also be dealt with. It applies when a computing device is stolen and therefore the TPM residing in the stolen device could be revoked.

Unfortunately, no suitable extension to the already known protocols was able to solve the problem convincingly. In fact a trusted credential transfer is not realisable with the current specification of the TPM. This can be demonstrated as either one of the following properties is essential. A credential must be issued and bound to either a TPM or a secret key of a user. In the second case, the credential is bound to a digital object which additionally needs to be bound to a hardware device. In the first case, TPM_0 must issue a credential to another TPM TPM_1 where the latter needs to be able to convince the verifier that it posesses a valid credential. TPM_1, however, is not able to pass the credential to another TPM. A solution could be to let TPM_0 issue credentials which is difficult as the signature by the issuer still needs to be present for the credential to verify. When bound to a user secret, the problem arises that there has to be a binding to the TPM as otherwise the misuse prevention would be categorised as purely software-based. The problem of both proceedings ended in a system which had the desired property but different usages of a credential ended up being linkable. Therefore, this approach has not been pursued further.

5.2.3 External Security Device

The range of available external security devices is broad and goes from PCMCIA-cards and USB-sticks to smart cards. The latter is one of the most resource-constrained ones and hence a good example to examine the feasibility of a misuse protection mechanism using an external security device.

Advantages

The biggest advantage over embedded security devices is that it is reasonable to assume a bijective mapping of devices to people. In particular, it is reasonable to assume one user to be holding one external security device per type of credential. For example, each user can have exactly one external security device from the Swiss government containing an electronic Swiss identification card. Additionally, each user might have a device from several other organisations. Notably, the user will not have two security devices from the Swiss government. Also, no two users will have the same security device for their credentials. Those two cases can arise when embedded security devices are used. Subsequently, there is no need to transfer a credential from one device to another one, which has shown to be still an unsolved challenge.

Disadvantages

A problem arising when taking an external security device is that it has more severe resource constraints, typically due to system constraints. For example, the bound on the computational power of passive smart cards with an RFID interface is constrained by the power that can be recovered using the remote interface. The operations still need to be performed fast in order to achieve a practical system. As an example, an authentication operation at the airport is tightly constrained in terms of the maximum time it may take. Deployment of a solution using an external device needs to be taken a look at. Even though not being a technical issue, a practical solution to these problems is crucial to the success of the system.

Another disadvantage is the size of the device which makes it easier to steal. This shows the importance of an appropriate mechanism to authenticate to the device. In particular, the assessment of a biometric authentication scenario would have been interesting. Due to the lack of an on-card reader for biometric data, we dismissed this subject. Furthermore, the possibilities of extracting the secured information are to be limited. Both challenges are out of the scope of this thesis and are being addressed already.

5.2.4 Conclusion

The use of an external device makes it needless to transfer credentials from one security device to another. This is enormously important as distinguishing between legitimate transfer and illegitimate transfer is an unsolved problem. In particular, the transfer from one device of one user to another of the same user does not have different properties from sharing or stealing a credential.

Because external security device render transfer of credentials unnecessary, we chose to use an external security device as opposed to an embedded device. The inconvenience due to the external device is, according to us, tolerable compared to the technical difficulties that arise from the use of an internal device. Another advantage lies in the trust model. As each organisation is able to issue its own security device, it does not have to trust the manufacturer of the internal security device. This is a huge advantage when it comes to real-world deployments of such authentication systems. Examples are smart cards used for authentication for e-banking. Each bank offers their own card for its system, which reflects exactly the above trust model arguments.

5.3 Biometry-based Misuse Protection

The simplest case of using biometrics is when an external token is protected with biometrics where the reader and evaluation is contained within the device. Recent research from [10] and [24] has shown that even key generation using biometric data is possible. This procedure could be used to even bind credentials to a specific user. Such a strong binding to a user is desirable for misuse prevention. The appearance of duplicate credentials in this scenario would imply the existence of a duplicate of the biometric properties of a user.

A similar proceeding is proposed in [18] where the biometric information is sent to the verifier through a warden. Even though those approaches are interesting we believe they are not yet mature enough to be implemented. We still want to mention those approaches as they might will prove to be useful in the future.

Chapter 6

A System for Effective Misuse Protection

A basic problem when dealing with duplicated credentials is that credentials are easy to copy as they are digital information. As seen in the previous chapter, this drawback of digital information can only be successfully dealt with when introducing a hardware-based protection mechanism. Thereby, some essential part of the credential is stored in a tamper-resistant device and it is secured such that extracting this essential part implies breaking the tamper resistance. The choice of an external device has already been motivated in Section 5.2. Especially, to use a smart card seems to be promising for a prototype implementation. As smart cards are very resource-constrained, they allow an interesting reasoning about the feasibility. Furthermore, smart cards can be manufactured at low cost which is important if the system should actually be deployed. A state-of-the-art smart card cannot be altered without the production of new hardware and is subsequently not suitable for our prototype implementation. However, a Java card offers the possibility to run applets which can be used to implement any functionality. The deployment of updates is particularly easy with Java cards as new applets can be installed instead of exchanging the whole hardware. Another argument in favour of the Java card was the availability at the IBM Zurich Research Laboratory, where the work for this thesis has been performed.

The binding to hardware, however, does not solve all the issues. A severe problem can arise if the tamper resistance of the hardware gets broken. The extraction of the secret should not result in a credential that can be arbitrarily copied and used. Hence, the detection of many instances of the same credential is to be targeted. A method to effectively reduce the number of copies of a credential would be able to prevent massive sharing. At the same time, a stolen credential could not be sold arbitrarily often. Two approaches that would be suitable are the K-show credentials and the counter-based credentials as outlined in Section 5.1. We chose the K-show credentials as they do not need the issuer to be online during authentication. Even so, there is a limit of K shows during one epoch which can by no means be surpassed.

Following the above argumentation, the misuse protection mechanisms that have been selected as the most promising are the following:

- K-show credentials, which assure that a credential is used at most K times during one epoch;
- Java card hardware protection assuring that a credential is only used in conjunction with a specified hardware device.

As already mentioned, the Java card has the desirable property of allowing very good deployment possibilities of updates. As a smart card, it is very resource-restricted and the feasibility of an implementation of an anonymous credential system with misuse protection has not been shown so far. The possibility to run applets allows the execution of arbitrary code which, however, will be running in the application layer.

The K-show credentials complement the hardware protection in an ideal way. Together with the counter-based credentials they are able to limit the incentive to steal credentials. As mentioned above, we preferred the K-show credentials over the counter-based ones, because the K-show credentials do not suffer from the disadvantage of needing an online issuer for showing a credential. In the following sections, we provide more details on the realisation of this effective misuse protection mechanisms.

6.1 K-show Credentials

The sharing prevention mechanism described in the paper by Camenisch et al. [3] is a cryptographic approach to limit massive sharing. Essentially, it reduces the number of times that a credential can be used to successfully and unlinkably authenticate at a verifier to K for each *epoch*, where an epoch is typically realised as a fixed-length time interval. Each credential is shown w.r.t. a token serial number (TSN) and additionally includes an cheater-identification number. During one epoch, the user can only make use of K TSNs. This comes as he has to prove in zero-knowledge, that a chosen number lies in the interval $[0, K[$. Therefore, he can only generate K different TSNs during one epoch. Consequently, in case the credential is shown more than K times, at least one token serial number (TSN) has been used twice. The cheater-identification number, on the other hand, assures the disclosure of the pseudonym of the user if at least one TSN is used twice. Thereby the verifier is able to extract the pseudonym which has been used during the issuance of the credential.

In the case of theft, the implementation of such credentials limits the number of times that a stolen credential can be sold. This comes as many copies will raise the probability of detection if there is no synchronisation and the users of the copies act independently. Thereby the incentive of an attacker to steal a credential is limited. A key point is the careful choice of the parameters K and the length of an epoch which determine the number of shared instances that can be around without being detected. Certainly, this number depends on the usage of the credentials. For example, a credential granting access to a newspaper is expected to be used several times a day whereas a credential holding the drivers license is rather shown seldom, say once a month. The parameters can be largely optimised as no general advice on their choice can be given.

The cryptographic details of the construction along with the assumptions can be looked up in [3]. Still, we present the most essential computations that need to be performed as well as the particular matching of epochs to real-time which has been chosen in our prototype implementation.

6.1.1 Credential Structure

The general structure of the credentials is taken from the *idemix* system. However, there are some additions to be made. Firstly, the fact that a credential is a K-show credential has to be encoded in the credential. Secondly, the value of K needs to be stored in the credential as well. We assume the maximal number of epochs to be $2^{l_{time}}$ and the upper bound on the choice of K to be $2^{l_{cnt}}$. Both parameters are provided as system parameters. Hence, those values do not need to be encoded in the credential. We have made this design decision as it does not restrict the generality of the approach to use the same upper limits on K and t_{epoch} throughout the system.

The essence of K-show credentials lies in the choice of the parameters K and the length of the epoch t_{epoch}. The system parameter l_{time} limits the number of epochs and subsequently it limits the lifetime of a credential. The latter can be computed as $2^{l_{time}} \cdot t_{epoch}$. As already mentioned, there is also an upper bound for the choice of K. Both boundaries are due to the design of *idemix*. All the same, the limitation should not affect the system design as both lengths are in the order of 30 bits. Subsequently, even a very small epoch length of one second would result in a credential lifetime of more than 34 years. Clearly, the maximal number of shows during one epoch has to be smaller than $2^{l_{cnt}}$ which implies the upper bound of $\sim 10^9$ on K. Subsequently, those limitations are restricting neither the choice of the epoch length nor the choice of K.

6.1.2 Issuance of Credentials

The receiver begins the issuance protocol by choosing a random value and committing to it. The commitment is subsequently sent to the issuer. The latter can add his own randomness in order to get a jointly-chosen random value. This randomness is used in the process of generating the TSN and the cheater-identification number in the holdership proof protocol. [3]

$$\begin{array}{ccc} \textbf{Issuer} & & \textbf{Receiver} \\ & \xleftarrow{C'} & s' \in \mathbb{Z}_q \\ & & C' = g^{s'} h^r \\ & & \\ \text{choose } s'' \in \mathbb{Z}_q & \xrightarrow{s''} & C = g^{s'+s''} h^r \\ C = g^{s''} \cdot C' & & s = s' + s'' \bmod q \end{array}$$

It is to mention that in the protocol above only the operations that differ from the currently implemented system are given in detail. The description of the whole system would go beyond the scope of this thesis and is in our view not necessary for the overall understanding of our extensions. A detailed description of the protocols is to be published by Camenisch and Sommer in [8]. The naming of the parameter in our protocols is in accordance with this paper.

6.1.3 Proof of Holdership of a Credential

Proving holdership of a credential is the action of showing the possession of a credential and this is subsequently often denoted “show protocol”. The protocol is carried out between two parties called *prover* and *verifier*. As the intuition suggests, the prover takes the role of showing his credential to the verifier. As explained in Section 4.3 by the term “show” we never assume that any attribute is revealed. More precisely, a basic instance of the show protocol proves to the verifier that the prover has, i.e. knows the content of, a credential from the given issuer.

Protocol Description

As already mentioned, the K-show credentials work by letting a user only generate K different TSNs during one epoch. The cryptographic realisation of this restriction is shown in the following protocol description.

Note that the subsequent protocol description outlines only the K-show extensions to the *idemix* show protocol. For the K-show extension protocol, the verifier chooses a random value which is forwarded to the prover. The domain of the randomness is to be large as only different random values allow the identification of cheaters. This argument will become clear in Section 6.1.4.

For a simplification of the protocol, we define a function $c(u, v, z)$. It allows the representation of the three values u, v and z as one bitstring and is defined as

$$c(u, v, z) := \left(u 2^{l_{time}} + v\right) 2^{l_{cnt}} + z.$$

This function visualises the necessity of restricting the maximal possible value for K and the epoch length. Additionally, we define a function $f_{g,s}(x) = g^{1/(s+x)}$ where $x, s \in \mathbb{Z}_q^*$ and g is a generator of group G which is of prime order q.

Along with the random value, the verifier provides the prover with the current epoch t. After having received those two values, the prover is able to compute the TSN[1] S and the cheater-identification number E. In addition to S and E, the prover needs to issue a zero-knowledge proof for a correct generation of certain values. Especially, he needs to prove that S and E are built correctly. Important is also the proof of knowledge of the jointly-chosen random value committed to in C and that actually the chosen value of j lies in the allowed range. This last statement is

[1]For a high-level description refer to section 6.1.

proven as the chosen value of γ is proven to lie in the specified range. Accordingly, the additions to the basic show protocol are given in the following protocol description. [3]

$$\begin{array}{ccc}
\textbf{Verifier} & & \textbf{Prover} \\
R \in \mathbb{Z}_q^* & \xrightarrow{t,R} & \begin{array}{c} S = f_{g,s}(c(0,t,j)) = g^{\frac{1}{s+c(0,t,j)}} \\ E = g^x h^y f_{g,s}(c(1,t,j))^R = g^x h^y \left(g^{\frac{1}{s+c(1,t,j)}}\right)^R \end{array} \\
 & \xleftarrow{E,S,\text{proof}} & \begin{array}{rl} PK\{(x,y,\alpha,\beta,\gamma): & S = f_{g,\alpha}(c(0,t,\gamma), \\ & E = g^x h^y (f_{g,\alpha}(c(0,t,\gamma))^R, \\ & C = g^\alpha h^\beta \\ & 0 < \gamma \leq K\} \end{array} \\
\begin{array}{c} \text{verify proof} \\ \text{store } (S;E,R) \end{array} & &
\end{array}$$

Epoch to Real-time Matching

In the case where only one verifier is used, the epoch choice is straightforward. The verifier chooses any epoch and the user simply has to show his credential w.r.t. the verifier-chosen epoch. Clearly, the choice of the epoch should be strictly increasing to relieve the users of the burden to save several counters. This would be necessary as the honest users want to comply with the restriction which only allows K usages during one epoch.

As soon as several verifiers are used for the processing of authentication requests, the choice of the epoch has to be synchronised among the verifiers. Practical systems show that synchronisation, e.g. time synchronisation, is a communication intensive task. Notably, it is unrealistic to assume that a system of distributed verifiers will be synchronised tightly. Therefore we want to relax the requirements for the synchronisation as much as possible. However, if a certain desynchronisation among the verifiers is allowed, the handling of epochs becomes more involved. Before this issue is discussed, we want to accentuate that we understand an epoch as a time interval and consequently the synchronisation refers to time synchronisation among the verifiers.

Using several verifiers in a completely unsynchronised way is not possible. Accordingly, we assume several verifiers to be synchronised such that no pair of verifiers is in two not neighbouring epochs. In other words, the maximal synchronisation error among all pairs of verifiers is smaller than the length of one epoch t_{epoch}.

Example 6.1.1. Let us call the current epoch that a verifier V_1 uses t_n. The epoch preceding t_n is subsequently called t_{n-1} and the succeeding epoch will be named t_{n+1}. As V_1 uses t_n at a certain moment, either all other verifiers will be using an epoch from the set $\{t_{n-1}, t_n\}$ or all of them are using an epoch in $\{t_n, t_{n+1}\}$.

The prover is allowed to show his credential w.r.t. the actual epoch of a verifier t_n or its preceding epoch t_{n-1}. The selection of the epochs on the verifier side makes it necessary to relax the requirements on the epoch a user can choose. Consequently, the verifier accepts proofs from his current and the immediate preceding epoch.

Example 6.1.2. On an authentication request of user U, the verifier supplies his epoch identifier t_n along with the challenge r. If the user is in epoch t_{n-1} and has not used the credential K times, he will not adapt his epoch and show the credential using epoch t_{n-1}.

This scheme of managing epochs has been chosen as it offers two advantages. Firstly, the verifiers do not need to maintain tight epoch synchronisation which is very useful as it limits the communication cost and leads to a vast simplification of the system. Hence, the system can be used even with a large number of verifiers. In particular, if the length of the epochs is chosen to be large (e.g. multiple hours or even days). Secondly, the user only needs to maintain one counter because he never needs to decrease the epoch. For example, if the verifiers were able to set the epoch to any value within a certain interval, the user would have to maintain a counter for each of the possible epochs. If the initial assumption on the synchronisation of the verifiers is met, there is no need for decreasing the epoch as the user is allowed to prove holdership w.r.t. the previous of the suggested epoch. Furthermore, there is some acceptance of heavy usage of the credential during one epoch. A credential can on average only be used K times per epoch. However, if the user uses his credential $k < K$ times during one epoch, he is allowed to use it $2K - k$ times in the succeeding epoch. For a better understanding of this advantage we provide an example. Additionally, in Fig. 6.2 the property is realised in an example.

Example 6.1.3. We assume verifier V_1 being in epoch t_n and verifier V_2 being in epoch t_{n-1}. Additionally user U currently uses epoch t_{n-1} having used the credential already $k < K$ times.

Upon an authentication request from U with V_1, the credential can be shown w.r.t. t_{n-1} and $k + 1$ only if $k + 1 < K$ holds. Otherwise the credential is shown w.r.t. t_n and K is reset to zero. In a successive authentication attempt, U contacts the same system but now he is being directed to V_2. Assuming that the epochs at both verifiers have not changed, U is required to authenticate using t_{n-1}. If $k + 2 < K$ holds, the user can proceed with the authentication. Otherwise the user has to wait until the epoch has changed.

If we assume k to be equal to $K - 2$, we exactly have a case where a prover is required to wait. One might argue that the user would have been able to authenticate successfully in both cases if in the authentication with V_1 the epoch t_n was used. This argument is correct and could be solved by using an approach where the prover simply uses the epoch provided by the verifier. Such a setting, however, forces the user to keep counters for all epochs that he could be supplied with, which increases the complexity. The concern should rather be addressed by the appropriate choice of the parameters K and t_{epoch}. In particular, they should be chosen such that an average user is not hindered.

As seen in the explanation above, there are two reasons for changing the epoch at the users side. Assuming the actual epoch of a user is t_n. The first reason is the authentication at a verifier using an epoch which is different from both t_n and t_{n+1}. The second is a the verifier proposing t_{n+1} in combination with the user having used his credential K times in epoch t_n.

We want to analyse the number of times a user can spend his credential during a certain epoch. To do so, we use the reasons for changing the epoch at the user's side. The second reason indicated above is particularly simple. Clearly, the user has spent his credential K times during t_n. The first reason is more involved as the number of credential usages in epoch t_n cannot be determined. Nevertheless, we can state that the user has shown his credential less than K times. The average usage per epoch is accordingly smaller or equal to K.

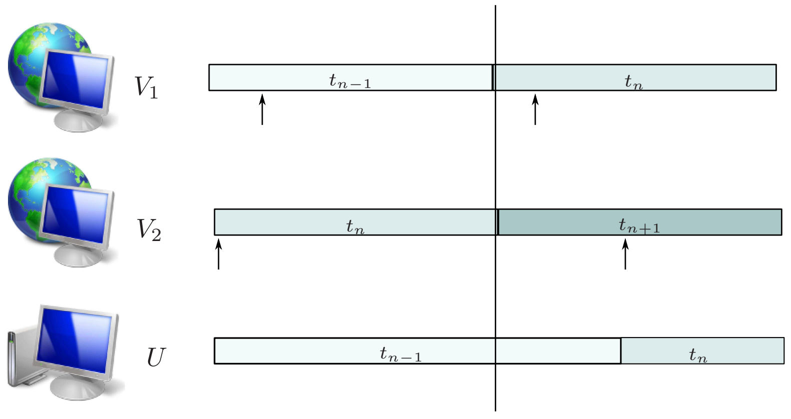

Figure 6.1. Illustration of the epoch adaption of a user U authenticating with entities having the maximal tolerable epoch desynchronisation. The arrows indicate authentications by U at the respective verifier

Example 6.1.4. Figure 6.1 shows how U would adapt the epoch assuming that he has not used the certificate already K times at any time. We can see that he is entitled to use epoch t_{n-1} longer than the actual length of an epoch. Thereby this mechanism guarantees that a user can actually authenticate K times during one epoch even though two verifiers occur to have the maximal tolerable desynchronisation.

In the case of perfectly synchronised verifiers, i.e., when only one verifier is used, the user benefits from a mechanism which we will call "burst tolerance". The mechanism also applies with several verifiers, if their desynchronisation relative to the epoch length is negligible. The benefit of the mechanism is that during one epoch length at the verifier, a credential can be shown at most $2K$ times. This seems to be contradictory with the claim that a credential can only be shown K times during one epoch. The solution of the supposed contradiction is the fact that an epoch at the prover's side can be scaled relative to the epoch length at the verifier's side. In other words, an epoch for the prover might seem to last shorter or longer compared to the epoch length at the verifier. The maximal expansion factor for prolongation of epochs is limited to 2, i.e., defining the epoch length at the verifier as t_{epoch}, the epoch length at the prover is smaller than $2t_{epoch}$. There is no lower bound on the length of an epoch at the prover's side. Thus, a user can make use of the allowed K usages in a time that is as short as he likes. However, after the usage of two epochs, i.e. t_{n-1} and t_n, he will have to wait for the epoch at the verifier to change. To illustrate this property, we provide another example as well as Fig. 6.2 which depicts both extreme cases of user-side epoch lengths in direct succession. Here, the shortening of the epochs is done such that the user can continuously authenticate at the verifier.

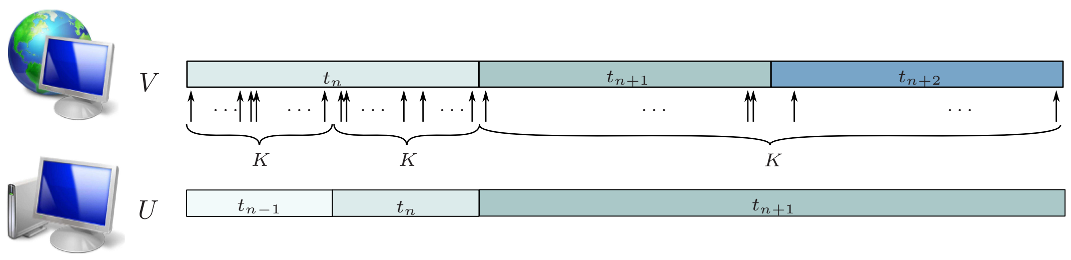

Figure 6.2. Illustration of the scaling that can happen if a user carries out many authentication attempts during a short period of time and afterwards only relatively few during another epoch. Note, that U would not be allowed to use his credential after the K usages w.r.t. t_n if not the epoch at the verifier had changed.

Example 6.1.5. Let us inspect a system with V using epoch t_n and U not having used his credential during epoch t_{n-1} and t_n.

U is informed during authentication that epoch t_n is used by V. According to the scheme defined above, he authenticates w.r.t. t_{n-1}. Clearly, he can authenticate K times using epoch t_{n-1}. At the authentication attempt $K+1$ he realises that no more authentications w.r.t. t_{n-1} can be carried out without being identified. Assuming that the epoch at the verifier has not yet changed, he increments his epoch counter to t_n. Still U is able to use his credential another K times as he is using it w.r.t. t_n. After U utilised the credential another K times, he will have to wait for the epoch at the verifier to be incremented. Consequently we can see that U has been able to use his credential $2K$ times even though the verifier only used one epoch identifier. This burst tolerance property can prove useful in a number of real-world applications.

6.1.4 Cheater Identification

The cheater identification is an important property of the overall scheme. It allows the verifier, or a delegatee, to revoke the anonymity of a user in case he overuses a credential. Overuse is given when the credential is used more than K times w.r.t. one epoch at the verifier[2]. The process is independent from the authentication process, which is essential for highly-loaded services. In

[2]Refer to the previous section for the exact semantics.

such systems it might even be tolerable to grant access to the system before having tested if the credential has been overspent. The anonymity can be revoked asynchronously.

The details of the cheater identification as described in [3], can be seen in the following computations.

$$\boxed{\begin{aligned} f_s(1,t,j)^R &= (E/E')^{(R-R')^{-1}} \\ \text{nym} &= \frac{E}{f_s(1,t,j)^R} \end{aligned}}$$

The only possibility for a user to overspend without being detected is, if the same value for R has been chosen. This is the reason for the domain of the randomness to be large. In a small domain the verifier might be unlucky and challenge the user with the same value which could be exploited by a malicious user. With a large domain we refer to a domain being of such a size that a collision, that is, the same randomness being chosen again, occurs only with negligible probability.

6.2 Java Card

We provide a short introduction to the design of smart cards and refer to [12] for more details. Smart cards have been around for decades and they are used in many scenarios where security-relevant data need to be stored or secure computations need to be performed. The tamper resistance of a smart card provides a certain level of security even in a hostile environment. The cards can be equipped such that they have only memory and are called *memory cards* or they come with an embedded microprocessor and are called *intelligent smart cards*. The memory cards need a way to perform access control to their data which is guaranteed by security logic functionality. The focus will be on intelligent smart cards as Java cards — our primary target — are equipped with a microprocessor.

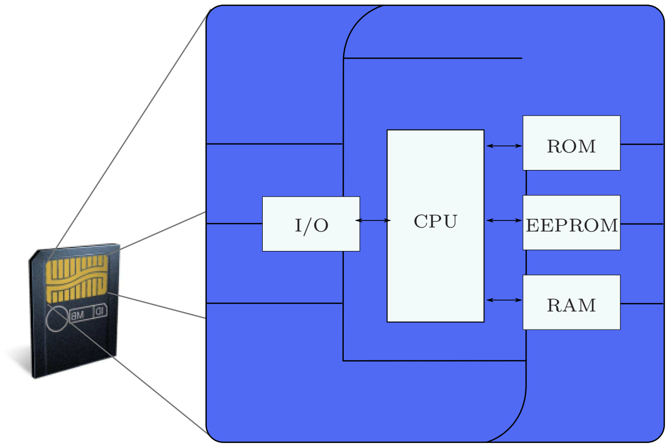

Figure 6.3. Architecture of a smart card with a microprocessor

The hardware that an intelligent smart card is based on, can be seen in Fig. 6.3. To compute the cryptographic functions efficiently, the card is usually equipped with a cryptographic co-processor. Security-relevant hardware features such as supply voltage control, frequency control, temperature sensor, read only EEPROM and so forth are provided.

The lack of strict boundaries between operating system and application layer requires any application to be written for a specific smart card only. Java cards were designed to overcome this problem. They implement a lightweight Java environment which is able to run on a smart card. The architecture of Java cards is explored in further detail below.

6.2.1 Architecture

Java cards are enhanced smart cards which are able to run a subset of the Java functionality. They run a virtual machine and a runtime environment much like the one needed for the execution of Java code on a usual computer. The biggest difference is that due to the resource restrictions the supported functionality is very limited.

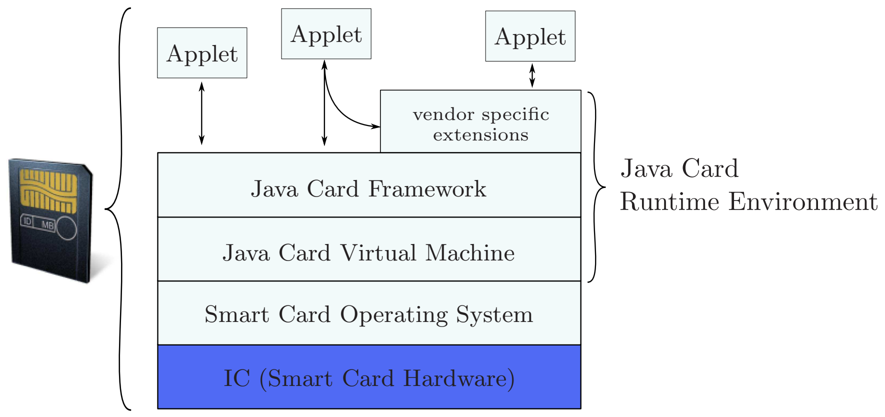
Applet
Applet
Applet
vendor specific extensions
Java Card Framework
Java Card Virtual Machine
Smart Card Operating System
IC (Smart Card Hardware)
Java Card Runtime Environment
ID
MB

Figure 6.4. Java Card architecture

Figure 6.4 shows the organisation of the Java runtime environment on a smart card. Some security-relevant aspects are worth being highlighted. First of all, the virtual machine (VM) retains its state in the non-volatile memory and therefore cannot be stopped by removing the power source. This is necessary as there is no internal power source in a Java card. The ability to stop the VM at in chosen state, however, could lead to security issues.

Second, there is a mechanism to separate applets from each other. It is called *firewall* and controlled by a a combination of the class loader and the access control functionality. This is an enhancement of the sandbox concept to the Java card runtime environment (JCRE).

Applets reside in their specific execution environment where they have access to other applets residing therein. Public methods are stored in a virtual heap implemented by the firewall. The access to applets in different execution environments is granted after a request to the JCRE has been forwarded to the specific applet and the latter grants access.

Memory management is essential as it is clearly security-relevant. In contrast to the standard Java environment there is no garbage collector. References to allocated memory therefore have to be handled with care. If values are needed during several sessions, the non-volatile memory is the only possible storage place. Some data however might only be needed for a short time and can be stored in random access memory (RAM). Such transient memory has to be requested explicitly at allocation time. The write time to non-volatile memory is considerably long (typically > 1.5ms) wherefore the allocation of RAM should be considered for non-security-relevant values. Moreover, object reuse is to be preferred over instantiation in order not to run out on memory.[12]

6.2.2 Functionality

As seen in the architecture of the Java card, a card offers only limited access to its hardware arithmetic. This is considered to be necessary in order not to expose security-relevant operations. On the other hand, the card offers a basic set of Java operations which allows the implementation of almost any functionality.

6.2.3 Communication

There are two possibilities of implementing the communication from a host to the smart card. The first communication is byte oriented the second one is block oriented where a block consists of several bytes. Block oriented communication can in the case of our Java card be carried in two ways. The first is via a contact interface using ISO 7816 standard or via radio frequency using ISO 14443. The first standard defines the physical interface, position of connectors and the electrical characteristics of the pins. The second one regulates the frequency range. Additionally they specify the format of commands between the communication partners as well as the communication protocols.

Data to be sent to the card is sent as application protocol data unit (APDU) as specified in ISO 7816-4. Especially, the Java card does not support extended length fields which defines the APDU that is sent from the host application to the card as follows:

CLA	INS	P1	P2	*Lc*	*DATA*	*Le*

CLA:	instruction byte where some values are used for card instructions and the like; values beginning from 0x80 should be used for application specific tasks
INS:	defines the instruction that it to be executed on the card
P1:	either the first parameter of the method that will be executed or an additional byte to define the instruction
P2:	either the second parameter of the method that will be executed or an additional byte to define the instruction
Lc:	length of the DATA field in bytes
DATA:	the data to be processed
Le:	length of the expected reply in bytes

The emphasized arguments *Lc*, *DATA* and *Le* are optional. The reply from the card to the host is structured similarly. Here, the data part is optional and its length is to be specified in the message with the *Le*-field. The trailer specifies the error code which the card will always send to the host. Hence, the status word, i.e., the error or success code, is sent in the obligatory fields SW1 and SW2.

DATA	SW1	SW2

6.2.4 Additional Properties

An open problem is the authentication at the card. The currently most used solution is to authenticate with a personal identification number (PIN). Clearly, there are better solutions to this problem, for example the use of biometrical information with on-card matching. This topic will not be discussed in more detail as it is out of the scope of the thesis and applies orthogonally to our results for misuse protection of credentials.

6.3 The *idemix* System

An implementation of an anonymous credential system by Dieter Sommer of the IBM Zurich Research Laboratory is called *idemix*. We added our proposed effective misuse protection mechanism, consisting of a software-based and a hardware-based mechanism, to *idemix*. A brief introduction of the design of the system is given here.

The *idemix* system is structured into two main layers: the protocol layer and the credential system layer. Figure 6.5 shows those two layers together with the underlying arithmetic. The message flow is indicated by arrows. Also, the inclusion of XML specifications is indicated. The arithmetic is depicted as a substantial speedup can be achieved when an *idemix*-optimised arithmetic is in place.

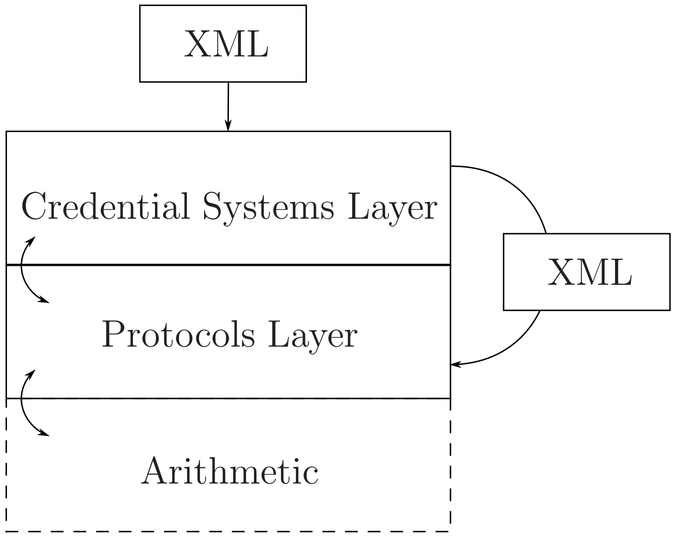

Figure 6.5. Overview of the *idemix* architecture

6.3.1 Credential System Layer

The credential system layer provides the high-level scenario of the credential system. Its purpose is to provide a clearly structured API for applications using *idemix*. In particular, the attributes of a credential are addressed in a straightforward way and the mapping to the attribute number in the actual credential is transparent. In other words, the credential system layer is responsible to map the datatypes that are provided in a credential to integer numbers. This is necessary as the protocols layer is only capable of handling integer numbers. For example, a long string can be mapped to several integers. Additionally, it realises the control flow on a high level of abstraction. For example, the abstract concepts such as pseudonyms or credentials are translated into cryptographic primitives. Ontology types and credential structure are known concepts for this layer. The protocols layer can not handle such concepts on an abstract level. However, it is essential that the credential structure can to be certified. To reach this goal, the structure is encoded into one integer attribute which can be passed on to the protocols layer.

6.3.2 Protocols Layer

The protocols layer implements the cryptographic protocols. As already mentioned, it only deals with integer attributes. The concepts it can handle are signatures, such as Camenisch-Lysyanskaya signatures, commitments and encryptions. For example, it is concerned with the effective message flow of the Camenisch-Lysyanskaya signature as specified in [7]. Similarly, it executes range proofs which have been previously specified by the credential system layer. The specification is written in an XML file to hand it over to the protocols layer. Also it can generate Pedersen commitments[3] simple discrete logarithm commitments. Just as well, it can issue zero-knowledge proofs which need to be specified using the XML structure known to the protocols layer.

[3]Refer to Section 3.2.3 for details.

Chapter 7

Prototype Implementation

After the determination of the effective sharing prevention mechanism, we implemented the approach. Certainly, some changes to the *idemix* system were necessary. Mainly the protocols layer was subject to changes. The more abstract credential systems layer needed to be made aware of the new credential types which include K-show credentials, smart card credentials or credentials which combine the use of the K-show limitation and the hardware binding.

7.1 Extensions to the *idemix* System

As mention before, the inclusion of the K-show credentials and the binding to the hardware requires some changes to the current protocols. The binding to hardware is widely transparent to the credential system layer. It is only reflected in the credential description and needs to be passed on to the protocols layer. Similarly, the use of K-show credentials is also given in the credential description to the credential system layer. The assignment of the latter is again limited to passing on the fact that the credential is a K-show credential along with the value of K. The details of the protocol changes are given in the following.

7.1.1 Integration of K-show Credentials

Given the introduction in Section 6.1, the protocol additions of the K-show credentials are not further discussed here. The insertion into the given system required some adaptions of the standard protocols. We consequently discuss only those changes.

The introduction of K-show credentials proved to be elaborate as the message flow, for example between prover and verifier, is defined. The addition of the jointly-chosen randomness in the issuing process fits well into the system. More difficult were the changes for the holdership proof. In particular, the defined message flow begins with a message from the prover to the verifier. Yet, for the K-show credentials the initiation of the protocols is done by the verifier by providing a random challenge in addition to the current epoch. This message is fundamental as it offers the randomness which is needed for the cheater identification process and the epoch proposition is essential for the choice of the epoch by the user.

As already mentioned, the current system unfortunately starts the proof at the prover's side. Simply adding a message before the currently used protocol would solve the issue. As this would make the integration into the Higgins Trust Framework[1] much more complicated, we did not regard this solution as appropriate. At this point we want to stress that the protocol implementation will be part of the open source Higgins Trust Framework. Thus we need to account for the restrictions imposed by the Higgins Framework.

Nevertheless, it is possible to solve the problem without changing the current message flow. The proposed solution works by replacing the random challenge using a hash, that is, we make use of

[1]Refer to `http://www.eclipse.org/higgins/` for more details.

an implementation of a random oracle. The choice of the epoch is done by the user. Certainly, the verifier still is in control of the valid epoch. If the verifier does not agree with the user-chosen epoch, he simply rejects the proof. In the case the user acts honestly, this situation arises because the epoch at the verifier has changed. The user, not being informed of the change, chooses an epoch which is no longer valid. This issue is solved by letting the user increment the epoch and start another authentication attempt. However, this user behaviour opens an attack vector. A malicious verifier could desynchronise the user to render his credential useless. The details of the attack are described in the following example.

Example 7.1.1. An honest user U authenticates to a malicious verifier V. The only thing the verifier does, is to reject each authentication attempt. Thereby, U increments his epoch repeatedly and the difference between the epoch used by U and the valid epoch, used by all honest verifiers, increases. In such a case, U would need to wait as many epoch as he has been desynchronised until he could successfully authenticate with a non-malicious verifier again. Probably the worst thing in the described attack is the fact that a desynchronised user would contribute to his own desynchronisation. This results, as his authentication attempt even with honest verifiers would fail and cause him to increment the epoch counter.

Assuming that the choice of the epoch is publicly known, this attack can be countered. For example, the epoch could be a certain time interval τ. The user still increments his epoch upon a rejected authentication attempt, but an additional requirement needs to be met. At least the time τ needs to have elapsed since the last epoch increment. Thereby assuring that a user can only be desynchronised by less than the epoch length. It assures that the user's epochs can only change as fast as the valid epochs determined by the honest verifiers.

Having dealt with the choice of the epoch at the user's side, we want to focus on the choice of the randomness. As mentioned, it is crucial to successful cheater identification. The user can certainly not be trusted to randomly choose a value. In particular, the repeated choice of the same number would disable the cheater identification, hence rendering the K-show per epoch limitation useless. A solution, secure in the random oracle model, is to choose a hash value instead of a random number. The input values to the hash function need to be predefined. The values consist of context information which is specific for each verifier and values computed for the proof of knowledge protocol that get known to the verifier during the protocol. The further makes sure that each verifier can detect a reuse of a specific R value, the latter provides additional input for the computation of the challenge. If there were no such restriction, the verifiers would need to coordinate the duplication check of the randomness over all verifiers. This would introduce enormous cost in communication between the verifiers and lead to a more complicated system architecture which would be impractical. In the proposed scheme each verifier can manage the checking of the randomness for duplicates itself. Especially, it needs to make sure no TSN[2] is used with the same randomness twice. Otherwise it risks to detect the overuse of a credential but not reveal the pseudonym used in the credential.

$$
\begin{array}{ccl}
\textbf{Verifier} & & \textbf{Prover} \\
 & & \text{select } t \\
 & & R = \mathcal{H}\left(url_v, \tau\right), R \in \mathbb{Z}_q^* \\
 & & S = g^{\frac{1}{s+c(0,t,j)}} \\
 & & E = g^x h^y \left(g^{\frac{1}{s+c(1,t,j)}}\right)^R \\
 & & PK\{(x,y,\alpha,\beta,\gamma): \quad S = f_{g,\alpha}(c(0,t,\gamma)), \\
 & \xleftarrow{t,R,E,S,\text{proof}} & \qquad E = g^x h^y (f_{g,\alpha}(c(0,t,\gamma)))^R, \\
 & & \qquad C = g^\alpha h^\beta \\
 & & \qquad 0 < \gamma \leq K\} \\
\text{verify proof} & & \\
\text{check for duplicate } R & & \\
\text{store } (S; E, R) & &
\end{array}
$$

[2]Token Serial Number as introduced in Section 6.1 on page 46

The accordingly-adapted show protocol can be seen above, where url_v is the verifier-specific value, i.e., context information. The second value of the function, i.e. τ, symbolises the other values from the show protocol which complete the hashed value. The changes regarding the original protocol are verifier-chosen epoch t and the computation of R.

The scheme requires the verifier to check the chosen randomness w.r.t. the database of authentications carried out during the actual epoch. This adds complexity for the verifier. On the other hand, the scheme allows to introduce K-show credentials without changing the defined message flow of the Higgins Trust Framework.

7.1.2 Integration of a Smart-Card-Binding

The advantages of a Java card for the implementation have been discussed earlier. Nevertheless, the incorporation of a smart card has a non-negligible disadvantage. As the card is supposed to be the only entity having knowledge of the secret key of the user, i.e. his MSK, the card has to perform all computations that involve the MSK. Outsourcing the computation would be possible in a secure way, if at least two non-colluding devices were present. In our case, we can only assume that one device, i.e. the host, is present. A technique for secure computational outsourcing is described by Hohenberger et al. in [17]. As already stated, the findings cannot be put to use as the host is the only outsourcing device present in our setting.

Consequently, we define that all operations using the MSK need to be executed on the card. Moreover, the secret should be information-theoretically hidden. We subsequently need the card to perform operations such as multiplication, addition and even exponentiation on arbitrary position integers. Such operations are available on the Java card but due to security restrictions they cannot be accessed from the within application layer. Considering this situation, we had to implement an arithmetic capable of computing all needed operations on arbitrary precision integers on the application layer.

The arithmetic has been implemented in accordance with the description of Knuth in [19]. Special attention needed to be paid to the modular exponentiation. This is a costly operation, wherefore we decided to implement an algorithm discovered by Montgomery. Used for an exponentiation, this specific algorithm causes at most twice as many Montgomery multiplications as the length of the exponent. The Montgomery multiplication itself is quadratic in the length of its arguments. The algorithms with better performance are far more complex and would not have been justifiable because of the relatively short arguments that have been used.

Special attention was needed for memory allocation. For example, once a portion of RAM has been allocated, it is only freed after the deselection of the applet or even the reinitialisation of the card. Consequently, the arithmetic allocates temporary variables which are needed repeatedly. Having to run the arithmetic operations in the application layer is a performance-limiting factor. However, there has been no other option to execute the necessary functionality. [12]

Programming an Arithmetic

Before actually implementing the arithmetic, we analysed the operations that would be necessary to embed the secret key into the Java Card. The sections below give more detail on those operations. The most important operation of the arithmetic in this context is the exponentiation. This results from the frequent use as well as the complexity of the operation itself. Other operations such as multiplication, addition, division and subtraction are clearly also needed.

We needed to implement the arithmetic as a Java Card applet. The Java code written can be optimised if the translation to byte code is kept in mind. Therefore, the order of declaration of variables as well as the number of loops had to be evaluated carefully. In addition to these techniques, variable reuse is a means which results in a speedup of the code. Unfortunately it also makes code much harder to understand. As the performance of the arithmetic is a key factor of the runtime of the whole protocol, even this method has been used to optimise the resulting byte code.

The implementation provides an arithmetic based upon 8 bit operations as the longest primitive on a Java card consists of 16 bits. The reason for not operating on 16 bit number is the carry. If 16 bit operations were used, the carry would be lost. The implemented algorithms follow the description given by Knuth in [19]. As already mentioned, the most involved algorithm which has been implemented was the Montgomery multiplication. This algorithm was adequate in implementation complexity considering the exponents we expect to occur. We can subsequently study the changes that need to be applied in the *idemix* protocols.

Adaptation of the Issue Protocol

As a first action, the credentials need to contain a value indicating that a smart card is used. The change of the XML specification and the parsing are straightforward and need no further description.

Issuing a credential includes binding it to the user. As the MSK of the user is now supposed to be secured by the card, the card will have to commit to the key during issuance. The information-theoretical hiding of the secret is currently done by the host. We show the operations executed by the card in the following protocol. As for the K-show credentials, we do not provide a detailed description on how a credential is built. Instead, our changes to the system are described.

$$\begin{array}{ccc}
\textbf{Host} & & \textbf{JavaCard} \\
 & \xleftarrow{C_x} & C_x = a_1^x \pmod n \\
U := C_x b^{v'} \pmod n & &
\end{array}$$

The commitment U is used for issuing the credential on. Subsequently, the host needs to proof to the issuer the fact that the commitment is built correctly. For this zero-knowledge proof some more computations are needed to be done on the card.

$$\begin{array}{ccc}
\textbf{Host} & & \textbf{JavaCard} \\
 & \xleftarrow{C_{r_x}} & C_{r_x} = a_1^{r_x} \pmod n \\
\tilde{U} := C_{r_x} b^{r_\xi} \pmod n & & \\
c_1 = \mathcal{H}(\ldots \parallel U \parallel \tilde{U} \ldots) & \xrightarrow{c_1} & c = \mathcal{H}(c_1 \parallel n_{card}) \\
 & \xleftarrow{c, s_x, n_{card}} & s_x = r_x + cx
\end{array}$$

This description corresponds to the current implementation. It can be seen that the secret x is not information-theoretically hidden. The reason being, that we tried to keep the computation time as small as possible. The entire computation of U on the card would result in information-theoretical hiding. This change of the protocol is straightforward to implement. Efficiency-wise it is to say that this change adds at most as many modular multiplications as the length of v' in the computation of U. The zero-knowledge proof would be computationally more intensive as well, where the additional modular multiplications are upper bounded by the length of r_ξ. In addition to the information-theoretical hiding of the MSK, this adaption would lead to a simpler protocol for the zero-knowledge proof. The simplification can be seen in the following protocol description.

$$\begin{array}{ccc}
\textbf{Host} & & \textbf{JavaCard} \\
c_1 = \mathcal{H}(\ldots \parallel U \parallel \ldots) & \xrightarrow{c_1} & \tilde{U} = a_1^{r_x} b^{r_\xi} \pmod n \\
 & & c = \mathcal{H}(c_1 \parallel \tilde{U} \parallel n_{card}) \\
 & \xleftarrow{c, s_x, n_{card}} & s_x = r_x + cx
\end{array}$$

However, there is another method to achieve a better hiding of the secret key from the host with the addition of a single modular multiplication. The protocol would be changed such, that the host sends the result of the computation of $b^{r_\xi} \pmod{n}$ to the Java card. The latter could multiply the received value with the result of $C_{r_x} = a_1^{r_x} \pmod{n}$ which would lead to $\tilde{U}$ without the host getting to know the value of the commitment C_{r_x}.

It is to say that the above protocols only describe the receiver's side. There need to be adaptions on the isser's side as well. Especially, the inclusion of the nonce of the card n_{card} needs to be taken into account.

Adaptation of the Show Protocol

The adaptation of the proof of holdership protocol is similar to the zero-knowledge proof in the issuance protocol. This follows from the prover proving knowledge of a randomised credential. The only part of the credential which is stored on the smart card is the MSK x. Subsequently proving knowledge of the credential from the point of view of the smart card is the same as proving knowledge of the MSK. The latter is done during issuance of the credential.

Still we want to indicate the necessary actions for the holdership proof on the prover's side. The verifier needs to adapt its protocols in accordance to the prover. Those changes are analogous to the extension on the prover's side with the key difference that the computations are performed on the same hardware.

$$
\begin{array}{ccc}
\textbf{Host} & & \textbf{JavaCard} \\
 & \xleftarrow{C_{r_x}} & C_{r_x} = a_1^{r_x} \pmod{n} \\
\tilde{U} := C_{r_x} b^{r_\xi} \pmod{n} & & \\
c_1 = \mathcal{H}(\ldots \parallel U \parallel \tilde{U} \ldots) & \xrightarrow{c_1} & c = \mathcal{H}(c_1 \parallel n_{card}) \\
 & \xleftarrow{c, s_x, n_{card}} & s_x = r_x + cx
\end{array}
$$

The host clearly needs to do more computations. In particular, it needs to prove knowledge of all attributes. Accordingly, it needs to provide the verifier with his random number n_{card} and the values of s_x and c. Those values are necessary to verify the zero-knowledge proof. The computations of the card ensure that the credential cannot be shown without the card computing the values as given above. An important role for this assurance is played by two challenges. The first comes from the verifier and is hashed into the value c_1 by the host. This challenge assures the freshness of the proof. The second, is the challenge used by the card n_{card}. It convinces the card that the value c is not chosen such that it allows the extraction of partial information about the secret x. The method of computing the hash value is performed in analogy as done in [1].

7.2 Measurements

We provide a running prototype of *idemix* using a Java card. In addition to that, the credentials can be issued as K-show credentials. Together with the secret key embedded in the smart card, the prototype offers an effective misuse protection.

The prototype implementation needed to be tested for its performance. As the arithmetic is placed in the application layer, it makes sense to give an expectation on the computation time, assuming the hardware arithmetic could be used. It is very difficult to make a reliable estimation as the computation time of very few operations are given. In addition to that, the details of the algorithms are not provided.

In addition to the reasoning about the performance of the prototype, we argue about memory requirements for extensions to the proposed system. Especially, we want to explore on extensions which make more extensive use of the external security device. As already mentioned, the current system has to be extended in order to hide the secret information-theoretically[3]. In addition to that, it would be interesting to have the most important credentials on the card. Thereby, the credential could be used with an arbitrary host as the card stores all the necessary values for the proof. However, in such a case the user either must trust the host he uses or the card will have to perform many more operations. In particular, the computations of the card need to be privacy preserving. It can be seen that the extension goes in the direction of running the whole *idemix* system on a smart card. That the current hardware technology is not very far from achieving that goal can be seen by the fact that *idemix* is running on mobile phones. As shown by Thomas Heydt-Benjamin the issuing takes less than 10 seconds and the proof of holdership can be executed in even less time.

We began the measurements by analysing the two protocols described in Subsection 7.1.2. The runtime is primarily caused by the exponentiations that are executed. Again, we want to mention that the arithmetic that executes all operations is running in the application layer of the Java card. Fortunately, the hash function is provided by the card. The runtime of the hashing can subsequently be neglected. Table 7.1 shows the measured runtimes for both protocols. Important to note are the system parameters which have been used. The length of base a_1, that is, the group size of the special RSA[4] group for the signature scheme, has been set to 72 bits. The length of the secret key is called l_x and has been set to 80 bits. It is used in one exponentiation in the issue protocol and accountable for the runtime difference between the two protocols. The most important value considering the runtime of both protocols is the length of the randomness used for the zero-knowledge proof. It is denoted as l_{r_x} and has been set to 344 bits. This is the shortest length which still allows the encoding of practical attributes. Certainly, those parameters are far too small for the system to be secure. Nonetheless, they are suitable for showing the feasibility of the prototype.

Protocol	t_{card}	$t_{emulated}$	Operations
Issue:	$\sim$ 9min 40s	$\sim$ 80s	$(l_x + l_{r_x})$ ModMult
Show:	$\sim$ 7min 30s	$\sim$ 60s	(l_{r_x}) ModMult

Table 7.1. Runtime for the issue protocol and the show protocol

The first column of Tab. 7.1 indicates the runtime for the execution of the issue and the show protocol on the Java card, respectively. The second column shows again the runtime of those two protocols with the difference that they are executed on a Java card emulator[5]. The last column of the table indicates the number of modular multiplications[6] which had to be performed. These numbers give an intuition on the runtime difference between the two protocols. They will be further used for the theoretical exploration in the next subsection.

Clearly, a system with a runtime of over 5 minutes is far from being practical. Therefore, the prototype cannot be used directly. However, the use of the hardware arithmetic would substantially increase the performance of the system. The expectations on the execution times are discussed in more detail in the next section.

7.2.1 Analysis of Computation Times

There are no specifications of runtimes for basic arithmetic operations on smart cards. Our results were attained by comparison with an operation whereof the runtime is known. Such an operation is the verification of an RSA signature. The Philips P8RF5016 smart card for example, specifies the runtime of this operation to be less than 400ms when using a 1024-bit group size. We use this hardware as the same card has been used as Java card whereon we analysed the runtime. Consequently, we will be able to estimate the speedup factor which can be achieved.

[3]Refer to section 7.1.2 for further details.

[4]Refer to Chapter 3 for the definition.

[5]The JCOP emulator has been used here. Refer to `http://www.zurich.ibm.com/csc/infosec/jcop_tools/entry.html` for details.

[6]Refer to [19] for details on the implementation.

The RSA signature verification operation is basically a simple exponentiation. In the case of a 1024-bit RSA signature verification this would be rising a 1024-bit base to a 1024-bit exponent. The maximal number of modular multiplications that are needed to perform this operation is equal to the number of bits of the exponent. Also, the algorithm causes at most as many squarings as the bitlength of the exponent. The length of the numbers to be multiplied is equal to length of the base. Assuming that a squaring and the multiplication are similar in their time intensity, the time of one modular multiplication $t_{modMult}$ of two 1024-bit numbers is calculated as follows:

$$t_{modMult} = \frac{400\text{ms}}{1024 + 1024} \approx 200\mu\text{s}$$

In this calculation we assume that the duration of the computations for each bit in the exponent is the same. In particular, we assume the duration to be independent of the value of the specific bit. This assumption does not hold in general. However, as there is a side-channel attack on the timing of the smart card, the security of smart card is enhanced by computing some operation if a bit of the exponent is zero. As a result the length of an exponentiation is independent of the number of ones in the exponent, i.e., the duration is only dependent of the overall length of the exponent.

As seen in Tab. 7.1 the number of modular multiplications for the issue protocol can be calculated by adding the length of the secret key l_x to the length of the randomness l_{r_x}. In addition to the modular multiplications, there are possibly squarings necessary which account for a doubling of the number of operations. This is a result of the squarings as well as the multiplications being dependant on the length of the exponent. For the *idemix* system, those parameter are suggested to be 594 and 592, respectively. The size of the base is suggested to be 2048 bit. Subsequently, the projected runtime of the issue protocol τ_{issue} can be estimated with:

$$\tau_{issue} \approx 2(594 + 592)4t_{modMult} \approx 1.8\text{s}$$

The length of the base scales the multiplication time quadratically. As the bases are twice as long as in the given RSA operation, the time for one multiplication is four times as long as in the RSA operation. On the other hand, we need to analyse the number of modular multiplications. This boils down to the analysis of the bitlength of the exponent. This length is indicated in Tab. 7.1 and the values, when taking secure system parameters, are discussed further above.

The same considerations as for the issue protocol can be made for the show protocol. This analysis is of even more interest as the show protocol is carried out more often. A runtime which is noticeable for the user is likely to be not acceptable for the show protocol whereas in the case of the issue protocol it might be tolerable.

Again, we assume the length of the randomness to be 594 bits. The bases, as described above, are of length 2048 bit. We explore the runtime for the show protocol, called τ_{show}.

$$\tau_{show} \approx 2(594)4t_{modMult} \approx 1\text{s}$$

We can see that even the slightly outdated smart card is capable of executing its part of the show protocol in an acceptable time. However, if the MSK is to be information-theoretically hidden, the computation time would be increased as another 2816 squarings and modular multiplications would be necessary. This is due to the random exponent which has to be chosen of this length. Similarly, the issue protocol would be extended by 2818 due to the information-theoretically hiding of the exponent and also by 2816 due to the randomness which is used in the zero-knowledge proof. As already mentioned, this zero-knowledge proof is computed in both protocols.

A function that is of interest and has not been considered in the explanations above is the hashing function. To get a reliable hash value, a part of the hashing needs to be done on the card. The necessity of this operation has been discussed beforehand and the length of the arguments depends on the system parameters. The measurements of the Philips smart card have shown that this card can perform hashing efficiently. This fact is shown in Tab. 7.2. The hashing of three different numbers of bytes has been analysed. The first row indicates the time for the hashing of two numbers of the indicated length. The second row shows the time that was needed for sending the value to the card, initiating the hashing and sending the results back to the host.

Number of Bytes	47	74	94
t_{hash}	6.1ms	10.1ms	10.2ms
$t_{overhead}$	272.4ms	274.2ms	310.2ms

Table 7.2. Runtime for the hashing operation

The analysis has been carried out as follows. We performed 10000 hashing operations followed by the computation of another 1000 operations. Thereby, we could extract the overhead caused by the communication between card and host and other operations. Subsequently, the approximate time for hasing the number of bytes could be computed. Notably, the runtime of 74 bytes is only marginally lifferung from the hashing of 94 bytes. This results as the hash function is padding its input to a multiple of 512 bits. Clearly, the 74 bytes as well as the 94 bytes input are padded to 1024 bit numbers.

7.2.2 Java Card Simulator

The overall runtime of the prototype is unfortunately not useful even when the system parameters are shrunk as much as possible. As the Java card emulator has a substantial runtime as well, we provide a software simulator. The simulator is building a so called virtual Java card. This card is running on the host hardware but it uses only functions that are available on the real Java card as well. In addition to that, the simulation environment allows the host to communicate with the virtual card as if it were a real Java card.

Although the virtual card uses the same arithmetic as the real card, the memory constrains are not applied. When adapting the Java card itself, we subsequently recommend the use of either a Java card emulator or the use of a real Java card. On the other hand, the correctness of Java card code can be best verified using the simulator. Testing is simplified as the native Java functionality can be used to check against the self-written code. Another advantage of course is the possibility to verify the functioning of the higher layer protocols.

The simulator basically implements the same functionality that is provided by Java card development tools such as the JCOP framework[7]. Clearly, our simulator does not implement the same range of functions. We restricted the simulator to the functions that have been needed so far. Essentially those functions are:

- initialisation of a terminal
- communication using application protocol data units (APDUs)
- initialisation of a card
    - applet loading
    - reading data from APDU
    - sending data to APDU
    - applet deselection
- Exception handling

Additionally, some functions needed to be implemented using the native Java library. For example, the Java card offers hashing functions but the API is not the same as the Java API. Therefore, we provide a function mimicking the API of the Java card hash function and in fact using the native Java implementation to perform the hashing. Such methods were for example the allocation of RAM, the generation of random numbers and the message digest, as mentioned above.

[7]Refer to `http://www.zurich.ibm.com/csc/infosec/jcop_tools/entry.html`

Parameter	EEPROM	ROM
MSK x		$l_x \approx 256$ bytes
Bases a_j	$il_n \approx 53 \cdot 256$ bytes $\sim$ 13kB	
Signature (A, e, v)	$\sim$ 460 bytes	
Attributes m_i	$\sim$ 20kB	

Table 7.3. List of memory requirements of credentials stored on a smart card

7.2.3 Memory Constraints

An analysis of the memory constraints can only be performed after determination of the size of a credential. Thereafter, we can argue about the possibility of storing credentials on the card. The memory constrains of the card will be analysed using the example of the Philips P8RF5016 smart card, which has been used throughout the thesis. Still, we would like to emphasise that a state-of-the-art smart card is equipped with more memory as well as a faster microprocessor. As an example, we would like to mention a specific smart card being able to run a lightweight webserver[8].

A credential consists basically of the signed attributes and the signature on them. To estimate the number of bytes which are necessary to store such a credential, we first have a look at the signature. It consists of the three values A, e and v. A, as a group element, consists of 2048 bits. The length of the exponent e is determined by the order of the chosen group and is in the order of 400 bits. The length of v is with the current system parameters in the order 2600 bits. The signature consequently accounts for somewhat less than 630 bytes.

The length of the attributes is much harder to estimate as the current implementation allows variable size of the attributes. As our aim is neither the determination of the precise size of a credential nor an upper bound on the size, we can go ahead with estimations. For example, a practically useful credential would consist of about 20 attributes. When assuming the average length of an attribute to be in the order of 50 bytes, we get the size of the attributes to be in the order of one kilobyte.

In addition to the credential-specific values, which have been described above, there are a number of system parameters which need to be contained in the card as well. Otherwise the card is not able to deliver all parameters needed for the proof of holdership of a credential. The bases of the attributes are an example for system parameters which need to be stored once for the whole system. As the bases are of the same length as the RSA modulus, they have length l_n. Typically this value should be in the order of 2048 bits for a decent security system. The number of bases is equal to the number of attributes. Moreover, the pseudonym requires two bases and the credential structure is encoded using one base. Subsequently, the system needs three additional bases. As the assumed average number of attributes is given above as 20, we want the limit on the number of attributes to be not less than 50. Consequently, there is a need for 53 bases which account for $53 \cdot 256$ bytes, i.e., about 13 kilobytes. Another parameter of interest is the secret key of the user. It can only be issued once for each card. This parameter, consequently, needs to be stored once for all credentials. Certainly, the lengths of different parameters need to be stored as system parameters as well. However, as there are only around ten such parameters, their memory requirements are negligible.

The memory constraints of the Philips P8RF5016 smart card are the following. The EEPROM is limited to 32kB, the ROM is 64kB and there are 2.3kB of RAM. With such constrains only few credentials could be completely stored on the card. Especially, given the assumptions from above and shown in Tab. 7.3, only one credential would fit on the card. However, this card is not a state-of-the-art smart card. Even though the newest generation of smart cards could be able to handle all the credentials of one user, this might not be a good thing to do as the value of the card would become substantial. Even though this scenario is appealing for the users, there might be a security risk associated with it. A careful analysis would have to be carried out before deployment of such a system.

[8] Refer to `http://www.gi-de.com/pls/portal/maia.display_custom_items.DOWNLOAD_SEEALSO_FILE?p_ID=5453` for more information.

When we explore the number of credentials that can be secured with one smart card, we consequently need to pay attention to the value of the sum of those credentials more than the memory required to store them. If this value gets too high, breaking the tamper resistance might become economically justified. The lack of trust among different organisations suggests the use of different cards. Subsequently, this lack of trust helps a user to not accumulate too much value on one single card.

The goal of managing entire credentials on the card remains. Especially, as there is the possibility to store entire credentials on one card but not user one card for all credentials of one user. For example, a new smart card could be issued for each high-value credential and the low-value credentials could be added to a card. Still, the trust relations between the issuing organisations need to be established. The benefit of this procedure is mobility, a credential gains. In particular, the user could use such credentials even when only a public host were available. However, the untrusted host could ask the card to reveal more attributes than actually needed. Hence, the privacy of the user would effectively be violated. To cope with this form of misuse, an input or output device could be used on the card. Using an input together with an output device adds to the possibilities of using the devices. For example, a screen could show which values are revealed and a keyboard could be used to agree with the transaction or cancel it. Still, the security device would need to perform many more operations. The feasibility of such a scheme would have to be shown first. Generally, it is a well-known problem that for obtaining certain security features, a trusted input or output device is required.

Chapter 8

Summary and Outlook

Beginning from a system as proposed in [5] and [20], we analysed the proposed mechanisms which might be usable for misuse protection. Exemplary mechanisms can be found in [3], [2], [5]. After the analysis of those mechanisms, we proposed a scheme for effective misuse protection in anonymous credential systems. Subsequently, we proposed a mechanism to cope with misuse of anonymous credentials and implemented it, based on the already existing *idemix* system. The resulting prototype has been analysed and conclusions were drawn. An overview over conclusions and an outlook on future work is provided in this chapter.

8.1 Summary

We have analysed a number of mechanisms which target the misuse of credentials in anonymous credential systems. Misuse can occur in many different cases. However, the misuse possibilities in a credential system can be extremely limited, if the use of duplicates of credentials is prevented. In a real world scenario, the absolute prevention is illusory. Nevertheless, a limitation on the possibilities of production and use of credential duplicates can be achieved. We propose that misuse which can technically be countered is associated with the use of duplicate credentials. Other misuse scenarios are, for example, that a user is threatened such that he is made willing to give away his credentials. This and other, similar scenarios have not been considered as they can only be countered to a certain degree by technical solutions.

In order to restrict the possibilities of creation of credential duplicates, a hardware-based protection approach is introduced. In particular, the credentials are bound to a smart card. The binding of credentials to a smart card can be attained by storing the master secret of a user in the tamper-resistant memory of the smart card. Consequently, the production of the first duplicate of a credential is associated with high costs. Those costs result from breaking the tamper resistance of the smart card. However, the production of many duplicates is not more costly compared to the production of only one duplicate. This is due to the fact that copying digital information is associated with only negligible costs.

We state that the incentive of actually breaking the tamper resistance of a smart card is only justified, if the costs caused by this action are exceeded by the value gained. In particular, an attacker will not execute an attack which introduces more costs than the expected revenue. Therefore, we can increase the efficiency of the misuse protection mechanism by limiting the number of copies of a credential that can be used simultaneously. As an example, if only a limited number of copies of a credential can be used, the attacker can not sell as many copies of a credential as he wants, without the duplicates being detected. To justify the attack, the value of one copy of the credential needs to be sold at a higher price compared to the case where an unlimited number of the same credential could be sold. Another successful approach would be the introduction of a counter associated with each credential. In addition to the counter, there is a signature on the counter that needs to be presented when proving ownership of a credential. Thus, different copies of a credential need to stay synchronised. As soon as a copy loses the current counter value and the signature, i.e. is no longer synchronised, the show protocol would fail. The synchronisation of the credentials introduces severe costs. As the simultaneous use of several copies of a credential is linked to communication costs, it makes the attack less likely to be profitable for an attacker.

The selection of one of those two approaches has been necessary as only a limited number of mechanisms could be implemented. In addition to that, we wanted to implement the most promising approach regarding the overhead of its protocols. All the same, we want to accentuate that not only these two mechanisms but also the other cryptography-based techniques could be used simultaneously. This is of great importance as some organisation might choose not to use the proposed mechanism but one of the other algorithms. As the complementary use of the described approaches is possible, they could be implemented as extensions to the system at a later point in time.

The choice among the two approaches which are able to limit the number of credentials has not been easy. They both have the capability to introduce some protection against theft. All the same, an average user should not be restricted in the usage of his credentials. This has been an important requirement for the system. Subsequently, the acceptable percentage of users that are limited by the misuse protection would probably have to be much lower than 5 percent.

The approach suggesting the introduction of a counter, being associated with each credential, works as follows. The value of the counter as well as a signature on the latter is needed for each proof of holdership. If a credential is copied, each copy would need to acquire the latest counter value as well as the signature to successfully be used. In other words, the copies need to maintain a synchronised state. This is generally believed to be costly. The main disadvantage of the approach is the necessity for the issuer to be online in order to sign the updated counter values after each successful execution of the show protocol.

The other approach introduces a limitation on the number of times that a credential can be shown during one epoch. This technique introduces a probabilistic mechanism for detecting duplicate credentials depending on their use. In other words, the independent use of many copies of a credential would eventually be detected because they would be used more than K times during one epoch. If the number of copies that can be used without being detected should be optimised, the copies need to maintain synchronisation. This results as the duplicates need to avoid showing a credential more than K times during one epoch. The implications for an attacker therefore are more drastic compared to the counter-based approach. Subsequently, we implemented the limitation of K shows during one epoch in addition to the hardware protection mechanism.

The prototype implementation is running partially on a smart card, which can be used together with a host in the process of acquiring a credential. Also, it can engage in a proof of holdership to prove the knowledge of a credential. However, as access to the hardware arithmetic of a smart card is not granted, we had to provide an arithmetic to execute the required operations. The provided 8-bit arithmetic is subsequently running in the application layer of the card. Therefore, the prototype is operating rather slowly. The execution of the credential issuance takes about 10 minutes and the ownership proof is not significantly faster. The prototype not being practical, we explored the expectable runtime when the smart card arithmetic could be used instead of our arithmetic. The estimated runtime was in the range of one second for the show protocol which is promising as it has been calculated for the slightly outdated Philips P8RF5016 smart card. The issuance protocol was estimated to take in the order of two seconds. According to us, the use of smart cards for misuse protection is not only feasible but very promising.

8.2 Outlook

The biggest challenge which has been solved in this thesis is the choice of an appropriate mechanism to protect organisations from misuse of credentials. This issue is of major importance before credential systems will actually be deployed. We believe that an appropriate mechanism has been selected. Especially, it is much stronger compared to the mechanisms that many organisations still rely on. However, the solution to one challenge as often caused new challenges to rise.

The first such challenge that needs to be addressed, is the choice of the parameters used in the K-show credentials. There is a major possibility for optimisation for each type of credential. According to us, the optimisation should be begun with the analysis of typical usage profiles. Such profiles are necessary to determine the lower bound on the choice of K/t_{epoch} as the average user should not be limited in the usage of his credential. The process could be continued by the analysis of other, more unusual user profiles. Other than that, the memory and communication requirements for different epoch length should be calculated. This would lead to an upper bound on t_{epoch}.

Another challenge is the implementation of the hardware-bound credentials with a smart card where the hardware arithmetic can be used. It would be interesting to compare the actual computation time of a real card with the projected times given in this thesis. In addition to that, a practical prototype could be attained. Although the proposed misuse protection system is believed to be efficient, there are still some optimisations or extensions waiting for a solution or the implementation, respectively.

Appendix A

Timetable

The project can be roughly divided into three parts: reading and understanding the theoretical part, designing and implementing the functionality and documenting both processes. A graphical overview should give an idea about the partitions of time that are used for specific parts.

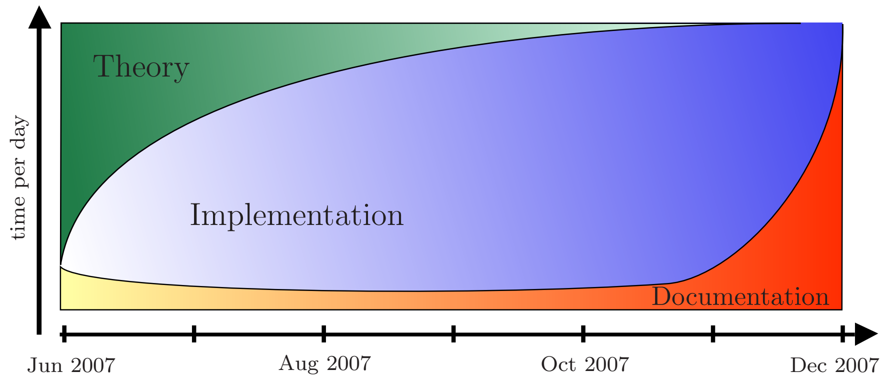

Figure A.1. Graphical overview over the project

The theoretical part consists of extensive reading into the subject of pseudonym systems at the beginning. It will be continually be more focused onto theft or misuse related topics of such systems. It will be mostly finished after the first month and builds the first milestone (M1) of the thesis.

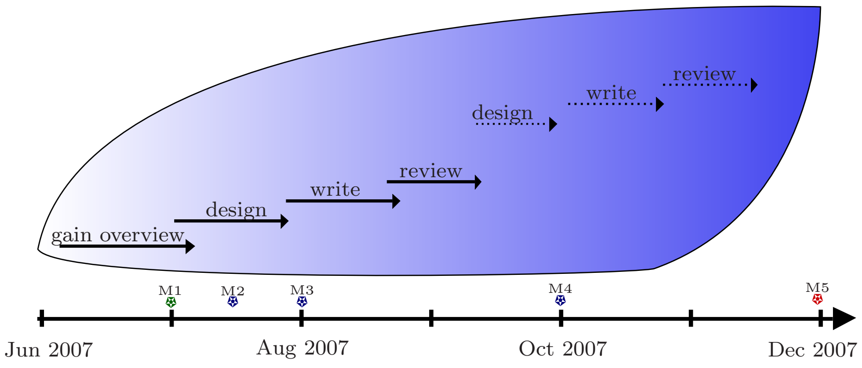

Figure A.2. Graphical overview over the implementation part including milestones

The most important part needs to be examined more closely. The programming is done by evolutionary prototyping meaning that a first version of the program is extended continually with functionality. Initially there will be needed some effort to get to know the architecture the other parts are repeatedly executed. Milestones are difficult to figure out however the raw design being the second milestone (M2) will be finished two weeks after the first milestone. The detailed design will be ready another two weeks later (M3). The last milestone of the implementation phase will be at beginning of October when a working implementation needs to be available.

The documentation part is kept at a reasonable level ensuring a complete documentation of the work will be finished by the end of the project (M5).

Bibliography

[1] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation, 2004.

[2] J. Camenisch. Protecting (anonymous) credentials with the trusted computing group's tpm v1.2. In S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, editors, *SEC*, volume 201 of *IFIP*, pages 135–147. Springer, 2006.

[3] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 201–210, New York, NY, USA, 2006. ACM Press.

[4] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash, March 2006.

[5] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Lecture Notes in Computer Science*, 2045:93+, 2001.

[6] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials, 2002.

[7] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks – SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2002.

[8] J. Camenisch and D. M. Sommer. – title unknown –. Description of the implementation of an anonymous credential system called *idemix*.

[9] J. L. Camenisch and M. A. Stadler. Efficient group signature schemes for large groups. *Lecture Notes in Computer Science*, 1294:410+, 1997.

[10] Y.-J. Chang, W. Zhang, and T. Chen. Biometrics-based cryptographic key generation. In *ICME*, pages 2203–2206, 2004.

[11] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[12] Z. Chen. *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[13] I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order, 2001.

[14] I. Damgård and J. B. Nielsen. Commitment schemes and zero-knowledge protocols, 2007. [Online; accessed 06-July-2007].

[15] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, pages 186–194, New York, 1987. Springer-Verlag.

[16] T. Heydt-Benjamin, C. Soriente, and B. Defend. Efficient cloning and splitting resistance for anonymous credentials and e-coupons. Short Paper.

[17] S. Hohenberger and A. Lysyanskaya. How to securely outsource cryptographic computations. In *TCC*, pages 264–282, 2005.

[18] R. Impagliazzo and S. M. More. Anonymous credentials with biometrically-enforced non-transferability. In *WPES '03: Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, pages 60–71, New York, NY, USA, 2003. ACM.

[19] D. E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*. Addison-Wesley Professional, November 1997.

[20] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. 1999.

[21] J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. C. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou, S. Guillou, and T. A. Berson. How to explain zero-knowledge protocols to your children. In *CRYPTO '89: Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, pages 628–631, London, UK, 1990. Springer-Verlag.

[22] N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Syst. J.*, 40(3):614–634, 2001.

[23] T. TCG. About the trusted computing group, 2007. [Online; accessed 15-August-2007].

[24] G. Zheng, W. Li, and C. Zhan. Cryptographic key generation from biometric data using lattice mapping. *icpr*, 4:513–516, 2006.