Self-blindable credentials with revocation

Wouter Lueks

Faculty of Mathematics and Natural Sciences
Master Computing Science and Master Mathematics

August 2011

Supervisors:
W.H. Hesselink
J.-H. Hoepman
J. Top

Abstract

The use of digital credentials is nowadays common-practice. However, the awareness of the privacy risk associated with their use has caused anonymous credentials to gain popularity over the last decade. Batina et al. showed that the self-blindable credentials from Verheul, which are intrinsically anonymous, can be efficiently implemented on smart cards. Unfortunately, the revocation of these self-blindability credentials is complicated.

In this thesis we examine some existing revocation schemes and propose a new self-blindable credential scheme that uses fast revocation. We define a formal model for the anonymity and unforgeability of such a credential scheme and prove the security of our scheme in this model. Furthermore, we show that the existing self-blindable credential protocols with revocation are not anonymous.

Preface

During a period of nine months I have worked as an intern at the Security group of TNO in Groningen to obtain two credentials that will not be revoked: a Master degree in Mathematics and one in Computing Science. The thesis you now have in front of you is the final result of this internship and embodies the last part of my education at the University of Groningen.

I would like to thank Jaap-Henk Hoepman from TNO, for being a patient but critical supervisor. He introduced me to the field of modern cryptography and taught me how to convert my intuitive arguments into rigorous proofs – a thing he will hopefully continue while supervising my PhD project. Furthermore, I would like to express my thanks to Jaap Top and Wim H. Hesselink, my supervisors from the University of Groningen for their precise reviews of this thesis as well as their support during my entire student career.

My time at TNO would not have been as enjoyable without my colleagues, both fellow interns as well as regular employees. The discussions at the lunch and coffee-table, work-related or not, were always interesting and inspiring. In particular, my thanks go to Gergely and Maarten for our constructive discussions about research and cryptography.

I owe a great debt of gratitude to friends for their continuing support and welcome distractions. A special word of thanks goes to Erik, Nynke and Paul for their wonderful help in writing this thesis.

Finally, I would like to thank my family and especially my parents for their support and understanding.

Wouter Lueks,
August 2011

Contents

Chapter 1

Introduction

In the last couple of decennia, credentials have become an essential part of our lives: to attend a match of your favourite soccer team you need a club card, to use public transportation you need your transportation pass, and to buy liquor you need to prove that you are over 18 by showing, for example, you driver's license. By itself these checks are rather harmless. However, the rise of pervasive digital technologies make this at threath to our privacy.

When you use your driver's license to prove that you are over 18 years old you simultaneously reveal your full name, your date of birth, your marital status, and the fact that you have a driver's license, none of which is pertinent to the case at hand. Without technological support the shopkeeper will be hard pressed to read and memorise all this information and will typically not write it down, therefore privacy is not at risk. On the other hand, when the driver's license is verified digitally the store can record this information. It can easily link this with your purchase information. In this case, the tell tale signs of the shopkeeper keeping a record are absent. We would like to design anonymous credentials that can be used instead of the ordinary ones, to mitigate these privacy issues.

A driver's license actually contains a number of facts about the holder. For the remainder of this thesis we define credentials to be these single facts, instead of the document as a whole:

> A credential is a trust provider's statement about a thing or a person that is relied upon by other parties [79].

For example, the statements "I am over 18 years", "I live in Groningen", "My first name is Wouter" and "I have a driver's license" are examples of credentials with the municipality Groningen as trust provider. The statement "I am a full-time Computing Science student" is an example of a credential with the University of Groningen as trust provider.

We focus our attention on credentials that are verified automatically, that is by a computer system, rather than a person. The entrance terminals to the public transport system are a good example of such a system. In the digital realm it is reasonably simple to make digital credentials that encode only a single statement. However, this does not solve all the privacy problems. As a matter of fact, since all verifications are now processed digitally it is much easier to store a record of them. This allows anyone with access to these records to aggregate them and to compile extensive sets of privacy sensitive information.

A first step towards an anonymous system would be to require that credentials are *anonymous*, i.e. given a digital credential it should not be possible to decide who owns it. This property can be achieved by identifying credentials only by a unique number instead of the holders name. This is how the anonymous public transport cards in the Netherlands are constructed. However, by examining records corresponding to this number, travel patterns of the holder of this card can be identified, even though the identity of the owner is unknown. Furthermore, these patterns may actually reveal the identity of the owner over time. Therefore, demanding untraceability is not enough, we need something stronger: *unlinkability*. Given two events it should be impossible to figure out whether they belong together. If a card is traceable then it is surely linkable, so unlinkability implies untraceability.

1.1 Smart cards

To make digital credentials usable they are often installed on so-called smart cards. These credit card-like cards feature a very small processor that can perform simple computational tasks. The Dutch public transport pass, the OV-Chipkaart, is an example of such a smart card. One can think of a smart card as a special purpose computer with a single task. In this thesis this task is to show a credential. Bank cards and credit cards often support other tasks as well.

It is often suggested to place the functionality of a smart card on the ever more present, and computationally much more powerful, smart phones. However, there are some compelling reasons to keep using smart cards. What they lack in computational power they make up for with security and robustness. They are very well protected against physical as well as digital attacks. This makes them the ideal keepers of very sensitive information, like personal identification numbers, corresponding to the credentials. An attacker needs these keys to be able to use your credentials, so it is paramount to keep them secret.

1.2 Anonymous credentials

There exist multiple cryptographic protocols that provide unlinkable credentials. Among them are complete systems like U-Prove [18] and Idemix [21]. Both offer a complete system for credential management. These protocols are constructed around a strong cryptographic primitive called zero-knowledge proofs, see Chapters 2 and 3. As indicated by the name they leak no knowledge, except for the fact that the holder has a valid credential, and therefore automatically guarantee anonymity.

However, their use has a high price. The resulting protocols are often too complex to be of practical use on smart cards. The protocol for showing a single Idemix credential already takes about 10 seconds [11] when implemented on a smart card, definitely too long for practical purposes. On the other hand, recent work by Mostowski and Vullers showed that U-Prove credentials can normally be shown in less than a second. These credentials are, however, linkable.

In 2001 Verheul demonstrated an alternative credential system [79]. He did not use zero-knowledge proofs to obtain anonimity. Instead, he used a new cryptographic operator, a pairing, see Chapter 3, to create a very special type of credential: self-blindable credentials. These credentials have the property that they can be completely transformed, i.e. blinded, each time they are shown. By using randomness in the transformation the owner of a credential can show a different credential each time, thus obtaining unlinkability almost for free.

An additional advantage of these protocols is that the blinding operation is cheap, which makes them much more appropriate for use with smart cards. In fact Batina et al. [5] demonstrated that such an anonymous credential can be shown in less than a second.

1.3 Problem statement

In real world applications we have to deal with fraud, theft and other possibilities that require invalidating a credential before it expires. Such a revocation operation is, however, not supported by many credential systems. We would like to extend the work by Batina et al. [5] with revocation, as their system, based on the credentials proposed by Verheul [79], is very efficient. However, revocation is especially hard to do for self-blindable credentials as they can be completely transformed each time they are shown. Contrary to earlier systems a credential cannot have any identifying number that is used for revocation checking, as this would immediately void unlinkability. This has led to the primary question of this research:

> How can we change the self-blindable credential protocol by Verheul [79] such that these credentials can also be revoked?

Another issue with the original self-blindable credentials as well as its more recent derivatives is that none of these have a security proof and are, in some cases, actually fundamentally flawed. Thus we obtain our secondary goal:

> Can we give a formal definition of the security requirements for anonymous credentials and can we find a protocol that is provably secure with respect to these requirements?

We managed to find satisfactory answers to both of these questions, as will become apparent in the next couple of chapters.

1.4 TNO

This master's thesis is the result of a nine month internship at TNO in Groningen. TNO, the Netherlands Organization of Applied Scientific Research (in Dutch: Nederlandse Organisatie voor Toegepast-Natuurwetenschappelijk Onderzoek) is an independent research organization with as primary goals to generate knowledge and expertise that can be used to assist and advise both companies as well as governments.

The security group at TNO studies the security and privacy aspects of digital systems. It consists of about 20 people, half of them works in Groningen, while the other half works in Delft. During my time at TNO I was part of this group. The collegues from the security group are involved in many different projects. These can be split into two types. The first consists of projects that are themselves not security related, but where security nevertheless plays an important role. One example is the assessment cloud computing as an alternative to ordinary server farms. Clearly, security is an issue here, although it is not the core problem.

The second type consists of directly security related projects. Still, the topics are quite broad. Some colleques work on new ways for doing risk assesment, while others are primarily involved in the development of new standards and compliance testing of these standards. Related to the latter is the creation of new corporate policies. On the technical side people are for example involved in penetration testing of existing systems, analyzing the security of existing protocols and determining which new technology is best suited for a given task.

To take part in new projects TNO needs to continuously keep and update its knowledge base. This means that sometimes research should take place that is not the immediate result of any project from an external source. TNO feels that the interest in systems which feature better privacy is increasing. This thesis is part of their effort to understand and develop such systems.

1.5 Reading guide

This thesis is constructed in such a way that it is accessible for anyone with some background in mathematics and computer science. This also means that experienced cryptographers will find the first couple of chapters to be superfluous. Table 1.1 aids the reader in selecting the chapters he/she should read based on their level of knowledge about cryptography.

To answer our research questions we first examine existing systems that support or can be used to support revocation in Chapter 5. We combine this with research into credentials and credential systems in Chapter 4. In Chapter 6 we examine self-blindable credentials in more detail and expose some flaws in the existing protocols. Finally in Chapter 7 we build our protocol and formally prove its correctness. This protocol is subsequently extended in Chapter 8. We conclude this thesis in Chapter 9 with some conclusions and suggestions for future work.

Basic math level	Begin in Chapter 2 for a brief introduction into the field of cryptography and to get a feeling for the types of techniques that we will use in the remainder of this thesis.
Basic crypto level	You can safely ignore Chapter 2, and instead begin reading in Chapter 3 which contains more modern cryptography that is essential to the protocols we will see.
Intermediate crypto level	Most of the theory in Chapters 2 and 3 will be known to you. In Chapter 4 you will find an introduction to credential systems in general, while Chapter 5 contains an overview of existing systems and how they use revocation.
Expert level	You can start reading in Chapter 6 which contains an overview of existing self-blindable credential techniques. In Chapter 7 we describe our protocol and the corresponding security proofs. Chapter 8 improves upon this protocol to make it more practical. Finally, Chapter 9 contains conclusions and suggestions for future work.

Table 1.1: This table contains a reading guide for the remainder of this thesis based on the experience of the reader. Note that this table is incremental, you start at your own level, and then continue through the remaining chapters.

For those readers with only little time to spare we advice them to examine the protocols in Chapter 7, while skipping the proofs, and Chapter 8, and to read the conclusion in Chapter 9.

Chapter 2

Cryptography Basics

To understand the remainder of this thesis the reader should be aware of some basic concepts and strategies used in the field of cryptography. This chapter aims to introduce these. If you already have an understanding of signature schemes, zero-knowledge proofs, security games and proofs of security in cryptography, then it is safe to skip to Chapter 3.

In this chapter we will use RSA as our primary example to illustrate encryption and signature schemes as well as to illustrate how to prove the security of cryptographic protocols. To achieve the latter we look into security games, cryptographic problems and reductions.

2.1 Public key cryptography

The need and wish to send secret messages is very old. A message, often called *plaintext*, can be kept secret by transforming it into a encoded form, the *ciphertext*, before sending it. This is called encryption. Let us follow the traditional naming scheme and call the sending party Alice and the receiving party Bob. Traditionally both Alice and Bob are trusted not to reveal the message. A third party, called Eve, takes the role of a malicious adversary who tries to eavesdrop the communication and recover the original message. This should be difficult for Eve given only the ciphertext.

When Bob receives the ciphertext, he transforms it back into the plaintext. This step is called decryption. Since Bob should be able to do this easily, in contrast to Eve, Bob needs to know more: he needs to know a secret. Before the invention of RSA by Rivest, Shamir and Adleman [71] in 1978 the only known solution was for Alice and Bob to share the same key[1]. The following example demonstrates a very simple substitution cipher in which a key is shared between Alice and Bob.

[1] Actually, Clifford Cocks at the Brittish intelligence agency GCHQ (Government Communications Headquarters), had already discovered this in 1973, but this internal document was kept classified until 1998 [74].

Example 2.1. Let $\mathcal{A} = \{\mathrm{A}, \ldots, \mathrm{Z}\}$ be the alphabet and $\mathcal{M} = \mathcal{A}^*$ the message space, i.e. all strings constructed from characters in this alphabet. A very simple encryption scheme would be to group the string into sets of 5 (this ensures that the length of the words does not leak information) and then permuting each individual character according to some permutation key. Consider the following permutation:

$$e = \begin{pmatrix} \mathrm{A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z} \\ \mathrm{C\ I\ H\ V\ Z\ Y\ X\ E\ F\ O\ B\ K\ Q\ A\ J\ U\ M\ P\ W\ D\ S\ T\ N\ L\ G\ R} \end{pmatrix}.$$

Using this permutation the plaintext

$$m = \texttt{THISI SACTU ALLYA REALL YBADC IPHER},$$

is transformed into the following ciphertext

$$c = \texttt{DEFWF WCHDS CKKGC PZCKK GICVH FUEZP}.$$

To decrypt the ciphertext the inverse of e is used.

The above is an example of symmetric-key encryption called a monoalphabetic subsitution cipher [6]. In all fairness, much better symmetric-key encryption systems exist, like AES [57]. However, even in good symmetric-key encryption systems both parties need to share a secret, the permutation key in the previous example. This approach has several problems. First, a secure channel is needed to transmit the secret key. Second, in a system with n parties, all wanting to communicate with each other, a total of $n(n-1)/2$ keys need to be securely communicated and stored.

In their 1978 paper Rivest, Shamir and Adleman demonstrated an alternative that solved the problem of key distribution [71] by proposing a system that has since become known as RSA. When Bob wants to receive a message he generates two keys: one public key and one private key. He publishes the former while keeping the latter secret. The private key is therefore often called the secret key. To encrypt a message to Bob, Alice uses Bob's public key. Bob on the other hand will use his private key to decrypt the ciphertext.

Before we can describe how RSA works we need to introduce some terminology.

Definition 2.2 (Algorithm [58])**.** An *algorithm* is a well-defined computational procedure that takes a variable input and halts with an output and is allowed to make probabilistic choices.

Unless otherwise specified an algorithm is assumed to run in (probabilistic) polynomial time. Every public key encryption scheme is of the following form.

Definition 2.3 (Public key encryption scheme)**.** Let $\mathcal{M}$ be the message space and $\mathcal{C}$ the ciphertext space. A public key encryption scheme consists of three algorithms: KeyGen, Encrypt and Decrypt.

KeyGen(1^k)**:** The algorithm KeyGen generates a public key $\mathcal{S}$ and a private key $\mathcal{P}$ of security level 2^k.

Encrypt$(\mathcal{P}, m)$**:** The algorithm Encrypt takes a message $m \in \mathcal{M}$ and a public key $\mathcal{P}$ and produces a resulting ciphertext $c \in \mathcal{C}$.

Decrypt$(\mathcal{S}, c)$**:** The algorithm Decrypt takes a ciphertext $c \in \mathcal{C}$ and a private key $\mathcal{S}$ and recovers the original plaintext.

This scheme is correct when for all security parameters k and all pairs $(\mathcal{S}, \mathcal{P})$ generated by $\mathsf{KeyGen}(1^k)$ the following holds:

$$\forall m \in \mathcal{M}: \qquad c = \mathsf{Encrypt}(\mathcal{P}, m) \quad \Rightarrow \quad m = \mathsf{Decrypt}(\mathcal{S}, c), \tag{2.1}$$

so any ciphertext produced by Encrypt can subsequently be decrypted by Decrypt.

The parameter k determines the security level that should be offered by the system. Roughly speaking a system that offers k bits of security takes approximately 2^k time to break. So a higher value of k means that the system is more secure. For complexity theoretical reasons KeyGen gets 1^k, the string of k ones, as input rather than k itself.

The RSA encryption scheme works over the ring $(\mathbb{Z}/n\mathbb{Z})^*$ for some value of n. To succinctly describe the scheme we need the following definition.

Definition 2.4 (Euler's totient function [44])**.** For a natural number n the totient $\phi(n)$ is the number of natural numbers less than or equal to n that are co-prime to n (i.e. have no divisors in common except 1). For p a prime number we have $\phi(p) = p - 1$. For $n = pq$ the product of two primes we have $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$.

The RSA encryption scheme is then defined as follows.

Definition 2.5 (RSA encryption scheme)**.** Let the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ equal $(\mathbb{Z}/n\mathbb{Z})^*$, where n is determined by the KeyGen algorithm. Then the algorithms are given by:

KeyGen(1^k)**:** The algorithm KeyGen first generates two primes p, q of approximately equal size such that $p = 2p' + 1$ and $q = 2q' + 1$ and p', q' are prime as well. Then it sets $n = pq$ and chooses an e such that $\gcd(e, \phi(n)) = 1$. The public key is given by $\mathcal{P} = (n, e)$. Next it determines the multiplicative inverse d of e modulo $\phi(n) = (p-1)(q-1)$, i.e. $ed \equiv 1 \pmod{\phi(n)}$. The private key is given by $\mathcal{S} = (p, q, d)$.

Encrypt$(\mathcal{P}, m)$**:** Let $\mathcal{P} = (n, e)$. Then the ciphertext obtained from the plaintext m is $c \equiv m^e \pmod{n}$.

Decrypt$(\mathcal{S}, c)$**:** Let $\mathcal{S} = (p, q, d)$. Then given the ciphertext c the message can be recovered by calculating $m \equiv c^d \pmod{pq}$.

The message space is rather limited. To encrypt larger messages one could split them into manageable pieces, but this does give rise to some vulnerabilities. How to adapt RSA to this practical situation is described in many tekstbooks on cryptography, see for example [58, Chap. 8] and [75, Chap. 11].

Theorem 2.6. *The RSA scheme defined in Definition 2.5 is correct with respect to equation* (2.1).

Proof. First we require a number of well-known facts from number theory which we will just state here. The order of the group $(\mathbb{Z}/n\mathbb{Z})^*$ is $\phi(n) = (p-1)(q-1)$, so by Langrange's theorem we have

$$x^{\phi(n)} \equiv 1 \pmod{n},$$

for all $x \in (\mathbb{Z}/n\mathbb{Z})^*$. Consider any key-pair $\mathcal{S} = (p, q, d)$ and $\mathcal{P} = (n, e)$. We know $ed \equiv 1 \pmod{\phi(n)}$, so we can write $ed = 1 + s\phi(n)$ for some $s \in \mathbb{Z}$. Consider an arbitrary message $m \in (\mathbb{Z}/n\mathbb{Z})^*$. For ciphertext $c = \mathsf{Encrypt}(\mathcal{S}, m) = m^e \pmod{n}$ we have

$$\begin{aligned} c^d &\equiv (m^e)^d \equiv m^{ed} \equiv m^{1+s\phi(n)} \pmod{n} \\ &\equiv m \cdot (m^{\phi(n)})^s \equiv m \cdot 1 \equiv m \pmod{n}. \end{aligned}$$

So indeed the RSA scheme is correct with respect to equation (2.1). □

The above proof only shows that the RSA scheme is correctly defined. It does not say anything about how hard it is to invert the encryption, that is, how hard it is to recover the ciphertext given the plaintext. In fact, the definition even allows the encryption and decryption functions to be identity functions on the message. We tackle this deficiency in the next section.

2.2 Proofs of security

To say we are secure against attacks we first need to specify which attacks we deem possible. For example, in the RSA scheme from the previous section we cannot hope to protect against an adversary that can easily factor the public modulus n. For if it could, then it could itself calculate the secret key. So the computational power of an adversary matters.

Now, consider the scheme in Example 2.1. Suppose an attacker can obtain an encryption of a single message of its choosing. He/she could then choose

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

to be that message. This choice would reveal the secret key that is being used. This type of attack, where the attacker can choose messages to be encrypted is called a chosen plaintext attack. So besides the attacker's computational power also its ability to interact with the system determines what is and what is not secure.

Finally, we should decide what constitutes a security break: what is the goal of the adversary? Does it want to be able to decrypt any given message, or does it maybe even want to recover the underlying secret key? Or is it content with something more mundane as being able to find out some information about a very specific set of ciphertexts? The last two notions are dealt with in the next section.

2.2.1 Security games

In the previous section we sketched an attack that would result in a total break: the attacker can easily find the secret key. Ideally, knowing a ciphertext would

not give an attacker any information about the underlying plaintext. In what follows we shall see how to transform this notion into a precise definition.

Suppose there is some leakage of information, then an attacker could conceivably detect this. We formalize this as follows. If some information is leaked then an attacker can create two message, m_0 and m_1, such that it can distinguish the encryptions $\mathsf{Encrypt}(m_0)$ and $\mathsf{Encrypt}(m_1)$. A system in which this is not possible (and clearly we desire such system) is said to have the *indistinguishability of encryption* property.

We also wanted to formalize how the attacker is allowed to interact with the system. The traditional method for describing these interactions is using a security game. Such a game gives a precise description of how and when an attacker can interact with the system. The goal of the attacker is to win the security game. It usually does so by breaking the security property, for example by being able to distinguish two messages while indistinguishability of encryptions was desired.

The following definition describes the game for indistinguishability of encryptions under a chosen-plaintext attack (IND-CPA). Notice how this definition combines both a security property, the indistinguishability of encryptions, as well as an interaction model, the chosen-plaintext attack.

Definition 2.7 (IND-CPA game)**.** Let $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a public-key encryption scheme. Then then the IND-CPA game is a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ that goes as follows.

- **Setup:** The challenger runs algorithm $\mathsf{KeyGen}(1^k)$ to obtain a public-private keypair $(\mathcal{P}, \mathcal{S})$. It provides the adversary with the public key $\mathcal{P}$ while keeping the private key $\mathcal{S}$ secret.
- **Find:** The adversary $\mathcal{A}$ produces two messages m_0 and m_1 on which it wants to be tested by the challenger.
- **Challenge:** The challenger generates a bit $b \in \{0, 1\}$ at random and sends $c = \mathsf{Encrypt}(m_b)$ to the adversary.
- **Response:** The goal of the adversary is to guess the value of bit b.

Since we are describing the IND-CPA game for a public-key encryption system, we do not need to explicitly mention that the adversary can make encryptions. Everybody knows the public key and can hence make encryptions. Because the adversary can choose the messages on which it will be queried, this gives a worst case estimate for the security of the protocol.

Even though we have used a rather weak notation of security, that is we only allow for an attacker that can make encryptions, it is trivial to win the IND-CPA game for the RSA variant described in Definition 2.5. The attacker can simply pick two different messages $m_0, m_1 \in \mathbb{Z}/N\mathbb{Z}$ and compute their encryptions $c_i = \mathsf{Encrypt}(m_i)$. Upon receiving the challenge c the attacker finds i such that $c_i = c$ and returns i. Since under the scheme of Definition 2.5 identical messages will encrypt to identical ciphertexts, this attack will always work. In fact any deterministic encryption scheme will be vulnerable to such an attack.

The above suggests that we need to introduce some randomization into our encryption schemes to prevent the chosen-plaintext attacks from succeeding.

The ElGamal encryption scheme [35] is an example of such a scheme that is not deterministic. Instead of working modulo some composite number n this scheme operates in a cyclic group G with generator g of prime order p. In this chapter we will just consider this to be an abstract cyclic group. In the next chapter we shall see some examples of such groups. The encryption of a message now also depends on a randomly generated ephemeral (i.e. short-lived) key.

Definition 2.8 (ElGamal encryption scheme)**.** Let the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ equal $G = \langle g \rangle$ a cyclic group of prime-order p as generated by the KeyGen algorithm. The algorithms are given by:

KeyGen(1^k): The algorithm KeyGen first generates a cyclic group G of prime-order p with generator g. Then it chooses a non-zero private value x (mod p) at random. The public key is given by $\mathcal{P} = (G, g, p, h = g^x)$, while the private key is given by $\mathcal{S} = (G, g, p, x)$.

Encrypt($\mathcal{P}, m$): Let $\mathcal{P} = (G, g, p, h)$. Then Encrypt first generates a random number k (mod p) that serves as an ephemeral key and sets $c_1 = g^k$. Next it calculates $c_2 = m \cdot h^k$. The ciphertext is the tuple (c_1, c_2).

Decrypt($\mathcal{S}, c$): Let $\mathcal{S} = (G, g, p, x)$. Then given the ciphertext $c = (c_1, c_2)$ the message can be recovered by calculating $m = c_2/c_1^x$.

A simple calculation shows that this scheme is indeed correct. Intuitively it seems that this scheme has a better chance of resisting chosen-plaintext attacks. We shall show that this is indeed the case, but first we need to describe how we can restrict the computational power of an adversary.

2.2.2 Computational power

For the purpose of this thesis we focus on adversaries with practical capabilities. This means that in general we assume that the computation power of such an adversary is polynomially bounded, although it is allowed to make random choices. Unfortunately, these basic assumptions on the computational power are in general not sufficient to prove the security of a system.

From the discussion in the previous section it might appear as if RSA encryption is completely useless. The main problem we discovered was that a message was always encrypted to the same ciphertext, hence allowing an attacker to obtain some additional information. Note however, that it still cannot invert the RSA-encryption in general, i.e. without prior knowledge about the possible plaintexts. In fact, this is widely assumed to be a difficult problem. This problem can be defined more formally as follows.

Problem 2.9 (RSA Problem)**.** Given as input an RSA modulus $n = pq$, an exponent e with $\gcd(e, \phi(n)) = 1$ and a random $c \in (\mathbb{Z}/n\mathbb{Z})^*$ find $m \in (\mathbb{Z}/n\mathbb{Z})^*$ such that $m^e = c \pmod{n}$.

The following definition states what it means for a problem to be hard.

Definition 2.10 (Hard problem)**.** A problem is *hard* if there is no probabilistic polynomial time algorithm that can solve the problem with a non-negligible probability.

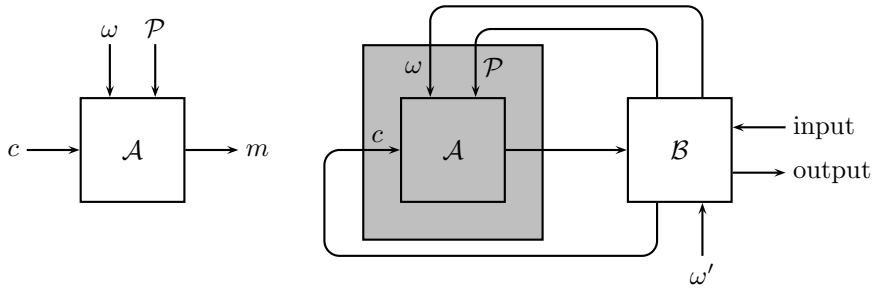


Figure 2.1: The adversary $\mathcal{A}$ gets as input some random string ω, the public key $\mathcal{P}$ and a ciphertext c and outputs the plaintext m. On the right we have a machine $\mathcal{B}$ that interacts with machine $\mathcal{A}$ to solve a difficult problem. Machine $\mathcal{B}$ gets its own random string ω' as input.

We will give a precise definition for non-negligible in the next chapter, but for now just assume that this means that a non-negligible event is not too rare.

So, as long as we assume that the RSA problem is hard, the core of the scheme still seems to offers some security. Indeed there is a variant of RSA, called RSA-OAEP, that can withstand chosen plaintext attacks, see for example [75, Chap. 20]. The main idea is to pad the message with random information that can only be removed while decrypting. The randomness ensures that identical messages can have different ciphertexts.

The above informal reasoning about the security of the core of the RSA scheme is very typical for security proofs. First, assume that some problem is hard to solve efficiently. Then use this assumption to argue that the system is secure. The applicability of these security proofs always depends on the validity of the assumptions that the proofs are based on. In other words, the proofs are relativistic.

Proofs of security almost always follow a fixed scheme. First, one assumes, to get a contradiction, that there exists an adversary $\mathcal{A}$ that can break the security of a system. Thus the adversary succeeds in winning the corresponding security game (with a non-negligible probablility). Then we build a machine $\mathcal{B}$ that can (ab)use the adversary to solve instances of the problem we assumed to be hard. Since this problem was assumed to be hard, such an adversary cannot exist, hence the protocol is secure. This scheme is demonstrated in Figure 2.1 for an adversary that can completely break an encryption scheme. Notice that our machine $\mathcal{B}$ has control over all the inputs of the adversary, including its randomness.

We can now show that the ElGamal scheme from Definition 2.8 is secure against chosen-plaintext attacks. We first define a suitably hard problem. Recall that if an adversary could factor the modulus n then it could recover the secret key. Something similar would happen if an adversary could calculate the discrete logarithm of $h = g^x$ with respect to the generator g. So as a basic requirement we require that the following problem is hard to solve:

Problem 2.11 (Discrete Logarithm (DL) Problem)**.** Let G be a cyclic group with generator g of order p. Then the Discrete Logarithm problem (DL) is to find for a given $h = g^x$, the discrete logarithm x of h with respect to g.

It turns out that this definition is not strong enough to prove the security of the ElGamal encryption scheme. However the following, slightly stronger problem does yield a proof.

Problem 2.12 (Decisional Diffie-Hellman (DDH) Problem)**.** Let G be a cyclic group with generator g of order p. Then the Decisional Diffie-Hellman problem is to decide, given g^x, g^y and g^z whether $xy = z \pmod p$. We call the tuple (g, g^x, g^y, g^z) a *Diffie-Hellman tuple* if $xy = z \pmod p$.

Note that if we can solve the discrete logarithm problem then clearly we could solve the DDH problem, so the former is indeed a stronger intractability assumption. Now we can prove the following theorem.

Theorem 2.13. *Assuming that the DDH problem is hard, the ElGamal scheme from Definition 2.8 is secure against chosen-plaintext attacks.*

Proof. Suppose there exists a polynomial time algorithm $\mathcal{A}$ that can win the IND-CPA game for ElGamal. We construct an algorithm $\mathcal{B}$ for solving the DDH problem. In the IND-CPA game $\mathcal{B}$ plays the role of the challenger. As input algorithm $\mathcal{B}$ gets a group G with generator g and a triple (g^x, g^y, g^z). Algorithm $\mathcal{B}$'s task is to decide whether $xy = z$.

Now algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows. It sets $h = g^x$ as public key and sends this to $\mathcal{A}$. Eventually $\mathcal{A}$ will produce two messages m_0 and m_1 on which it wishes to be challenged. Now we somehow need to harness the power of $\mathcal{A}$ in distinguishing encryptions of m_0 from those of m_1. Algorithm $\mathcal{B}$ does this by first choosing a bit $b \in \{0, 1\}$ at random. Then it sets

$$c_1 = g^y \qquad \text{and} \qquad c_2 = m_b g^z$$

and sends these to $\mathcal{A}$. We argue why is this a good choice on a case by case basis. If indeed $xy = z$ then $g^z = (g^x)^y$ and hence (c_1, c_2) is a valid encryption of m_b. Therefore, $\mathcal{A}$ will decide correctly with non-negligible probability. If on the other hand $xy \neq z$ then with overwhelming probability (c_1, c_2) is invalid, hence the output of $\mathcal{A}$ will be either true or false with equal probability.

Therefore, algorithm $\mathcal{B}$ outputs true whenever $\mathcal{A}$ decided correctly and false otherwise. We can make this algorithm better by running machine $\mathcal{B}$ multiple times. Hence we have a probabilistic polynomial time algorithm for solving DDH, which we assumed was not possible. Hence the ElGamal scheme is secure against chosen-plaintext attacks. □

In the above we were a bit imprecise about the success probability of algorithm $\mathcal{A}$ in order to keep the proof simple. The proof can easily be augmented to deal with this correctly. The next chapter contains a number of examples.

2.3 Digital Signatures

By signing a contract using a handwritten signature you declare that you agree with the content of this contract. In general it is hard for somebody else to fake your handwritten signature. This is important for two reasons. You are protected against others posing as you and the receiver of the contract knows that you cannot claim you did not sign it.

Digital signatures are the digital counterpart to these handwritten signatures. Contrary to traditional signatures, however, they also depend on the message. In general, signatures on two different messages are not identical. We will focus on signature schemes that use public-key cryptography. Signatures allow two basic operations: signing and verifying. Since signing should be hard, the signer uses his/her private key to produce a signature on a message. Verification on the other hand should only require the public key as everybody should be able to verify a signature.

Just as with public-key encryption schemes we can give a definition of a public-key signature scheme.

Definition 2.14 (Public key signature scheme)**.** Let $\mathcal{M}$ be the message space and $\mathcal{H}$ the signature space. A public key signature scheme consists of three algorithms: KeyGen, Sign and Verify.

$\mathsf{KeyGen}(1^k)$: The algorithm KeyGen generates a public key $\mathcal{S}$ and a private key $\mathcal{P}$ of security level 2^k.

$\mathsf{Sign}(\mathcal{S}, m)$: The algorithm Sign takes a message $m \in \mathcal{M}$ and a private key $\mathcal{S}$ and produces a digital signature $\sigma \in \mathcal{H}$ over the message m.

$\mathsf{Verify}(\mathcal{P}, \sigma, m)$: The algorithm Verify takes a signature $\sigma \in \mathcal{H}$ on a message m and a public key $\mathcal{P}$ and verifies that σ is a valid signature on the message m corresponding to the public key $\mathcal{P}$. If the signature is valid it returns True and False otherwise.

This scheme is correct when for all security parameters k and all pairs $(\mathcal{S}, \mathcal{P})$ generated by $\mathsf{KeyGen}(1^k)$ the following holds:

$$\forall m \in \mathcal{M}: \qquad \sigma = \mathsf{Sign}(\mathcal{S}, m) \quad \Rightarrow \quad \mathsf{Verify}(\mathcal{P}, \sigma, m) = \mathsf{True}, \tag{2.2}$$

so every signature produced by Sign should classified as valid by Verify.

As a first example we consider a variant of the RSA encryption scheme we saw earlier. The trick to obtain a signature scheme is to reverse the order of operations: to sign a message m we 'decrypt' it, while we 'encrypt' it for verification. This works, because encryption and decryption can be reversed in RSA. We end up with the following complete scheme.

Definition 2.15 (Basic RSA signature scheme)**.** Let the message space $\mathcal{M}$ equal $(\mathbb{Z}/n\mathbb{Z})^*$. The algorithms are as follows.

$\mathsf{KeyGen}(1^k)$: The algorithm KeyGen first generates two primes p, q of approximate equal size such that $p = 2p' + 1$ and $q = 2q' + 1$ and p', q' are prime as well. Then it sets $n = pq$ and chooses an e such that $\gcd(e, (p-1)(q-1)) = 1$. The public key is given by $\mathcal{P} = (n, e)$. Next it determines the multiplicative inverse d of e modulo $\phi(n) = (p-1)(q-1)$, i.e. $ed \equiv 1 \pmod{\phi(n)}$. The private key is given by $\mathcal{S} = (p, q, d)$.

Sign$(\mathcal{S}, m)$**:** Let $\mathcal{S} = (p, q, d)$. The signature on the message m is given by $\sigma \equiv m^d \pmod{pq}$.

Verify$(\mathcal{P}, \sigma, m)$**:** Let $\mathcal{P} = (n, d)$. The signature σ over m is valid if and only if $\sigma^e \equiv m \pmod{n}$. If it is valid the algorithm returns True and False otherwise.

The correctness of this scheme follows easily from the correctness of the related encryption scheme. Just like with encryption the message space is limited. To create a signature over a larger message we could apply the scheme multiple times, but apart from security issues this also has an additional conceptual problem that was not present for the encryption scheme. It does not match with our idea of a signature as something that is small and preferably independent of the size of the message we sign. For signature schemes this can be elegantly solved by cryptographic hash functions. Such a function maps a message of any length into a fixed sized co-domain. A signature on a message can then be replaced by a signature over the hash of the message. The following definition of a secure hash-function ensures that this adaptation does not impair the security of the overall signature scheme.

Definition 2.16 (Secure cryptographic hash function [75])**.** A function $H : \{0,1\}^* \to \{0,1\}^k$ is called a *secure cryptographic hash-function* if it satisfies the following three properties:

Preimage resistance Given a value $h \in \{0,1\}^k$ from the co-domain of the hash-function, it is hard to find a string x such that $H(x) = h$.

Collision resistance It is hard to find two messages m_1 and m_2 such that $H(m_1) = H(m_2)$.

Second preimage resistance Given an arbitrary message m_1 it is hard to find a second message m_2 such that $H(m_1) = H(m_2)$.

The three properties guarantee, in order, that an attacker cannot cook up a message for which it then knows the signature, that the signer cannot later repudiate his/her signature by giving another message, and finally that an attacker cannot obtain a signature on a message other than the one you just signed. Details for these attacks can for example be found in [75, Chap. 14].

Now that we have a definition of a signature scheme we can think of the security properties we want such a scheme to have. Just like with encryption schemes the adversary can have different goals. For signature schemes we usually distinguish between the following three.

Total break The adversary is able to recover the private key, thereby completely breaking not only the signature scheme, but also any other scheme that depends on the secrecy of the private key.

Universal signer The adversary is able to sign any message, without necessarily knowing the private key. This break is a bit weaker then the previous, since other schemes using the private key are not necessarily broken.

Existential forgery This is the weakest form of attack. The adversary is able to forge a signature on a message of its own choice.

As with encryption schemes we want even the weakest goal to be too hard, so in this case we do not even want existential forgeries. Similarly, we can analyze the different types of interactions. The weakest form would be for the adversary to produce a forgery without any assistance, this corresponds to the chosen-plaintext attacks on encryption schemes. For signature schemes we prefer to allow a much stronger attack notion in which the adversary may make any signature request it wants, but has to return a forged signature on a message for which it did not request a signature before. This is captured by the following security game.

Definition 2.17 (The EUF-ACMA game)**.** Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be an public-key signature scheme. Then then the EUF-ACMA game is a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ which proceeds as follows.

- **Setup:** The challenger runs algorithm $\mathsf{KeyGen}(1^k)$ to obtain a public-private keypair $(\mathcal{P}, \mathcal{S})$. It provides the adversary with the public key $\mathcal{P}$ while keeping the private key $\mathcal{S}$ secret.
- **Signature Queries:** The adversary $\mathcal{A}$ makes signature queries on messages $m_1, m_2, \ldots, m_q$ to the challenger. The challenger answers these queries in sequence. The adversary is allowed to be adaptive, so message m_i may depend on the responses to the queries for $m_1, \ldots, m_{i-1}$.
- **Output:** The adversary outputs a signature σ on a message m that was not queried during the previous phase, so $m \neq m_1, \ldots, m_q$.

To prove that a signature scheme is secure against adaptive chosen message attacks is very difficult. Recall from the previous sections that to make a reduction proof we need to make an algorithm that can answer all the queries from the adversary, even the signature queries. In general this is difficult. In the next chapter we shall see two examples for solving this problem. The first uses the fact that hash functions should behave as random functions, this idea is called the random oracle model. The second uses a stronger intractability assumption.

2.4 Zero-knowledge

Knowing something that others do not is often important in cryptography and its applications. For example, because only Alice knows her private key others trust that signatures corresponding to her public key are really signed by Alice. This also means that her secret, in this case her private key, can be used to authenticate her. If Alice can somehow show that she knows this private key then clearly she must be her. Suppose Alice wants to authenticate herself with Bob. Then they could run the following very simple protocol:

1. Bob generates a random nonce n and sends it to Alice.
2. Alice signs the nonce n using her private key and sends the result back to Bob.
3. Bob checks the signature against Alice's public key.

At first sight this protocol appears to be secure. The only way for another person to pose as Alice would be to fake the signature, which is something that should be very hard for a good signature scheme. However, in this scheme it

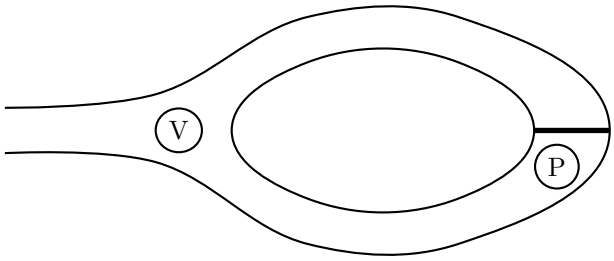


Figure 2.2: The cave used to illustrate the story of Ali Baba. In this picture Victor (V) shouts to Peggy (P) through which passage she should exit the cave.

is not Bob that is vulnerable to attack, but Alice. Since, if Bob is malicious, then he could obtain a signature on any message he wishes.

The problem in the above protocol is that Alice provides more information that she actually wanted to reveal, and this information can be abused. Modern cryptography offers an elegant solution: zero-knowledge proofs of knowledge. In these proofs no additional information is leaked apart from the fact that the assertion is true.

But how can you show that you know something without revealing anything about what your know? This seems counter-intuitive. Quisquatar et al. gave a very elegant example to explain the concept of zero-knowledge proofs of knowledge [70]. The story recounts the rediscovery of Ali Baba's cave in recent times. The cave in this story has a single entrance, which then splits into two passages. At the far end of the cave these two passages are connected by a hidden door, see Figure 2.2. This hidden door was used by Ali Baba to evade pursuers. Only when the secret passphrase, "Open Sesame", is spoken, does the door open.

Suppose that Peggy, the zero-knowledge cousin of Alice, tries to prove to Victor that she knows the secret passphrase. Of course this is valuable information, so she will not provide it to Victor so he can verify it for himself. As an alternative they play the following game. First both inspect the cave, then Victor waits outside the cave while Peggy enters one of the passages. After a while Victor walks up to the fork an shouts to Peggy from which side she should arrive. If Peggy really knows the secret passphrase she can always succeed in meeting the challenge. If not, she will fail half of the time. By repeating this experiment a sufficient number of times Victor can be convinced that she really knows the secret. Incidentally, this is a good example of a cryptographic protocol; a specification for how Peggy and Victor should interact to achieve a goal. More formally we have the following definition.

Definition 2.18 (Cryptographic protocol [58])**.** A *cryptographic protocol (protocol)* is a distributed algorithm defined by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective.

Indeed Peggy and Victor execute a proof of knowledge, as Peggy proves to Victor that she knows the secret. But how can we show that Victor learns

nothing besides the fact that Peggy knows the secret? Suppose Victor wishes to convince Claire that he knows that Peggy knows the secret. To this end he decides to wear a hidden camera and makes a recording of the whole experiment. Claire, however, is not impressed . For all she knows Victor hired an actor to play the role of Peggy and told her in advance in which sequence he would challenge her.

The fact that Victor could have simulated the entire experiment is at the heart of the zero-knowledge part of this zero-knowledge proof of knowledge. For if Victor could simulate the whole experiment without Peggy then clearly he did not learn anything by performing the protocol with Peggy.

Chapter 3

Advanced Cryptography

This chapter develops some more advanced concepts from cryptography that are necessary in the remaining chapters of this thesis. In this chapter we also explain the notation used in the remainder of this thesis.

First we fix some general notation in Section 3.1. In Section 3.2 we give a formal definition of zero-knowledge proofs including examples of how zero-knowledge proofs for discrete logarithms work. These zero-knowledge proofs will be at the heart of many protocols we examine in Chapter 5. Self-blindable credentials are based heavily on pairings, a mathematical operator that is defined over elliptic curves. We study them in Sections 3.4 and 3.5. A number of signature schemes are used in the construction of various self-blindable credential schemes. We study these in Section 3.6. Finally, to prove correctness of some of these signature schemes we need to work in the random oracle model. We explain this model and prove some important results about it in Section 3.7.

3.1 Notation

In this section we will describe some notations and conventions that are used throughout this thesis. We use the notation $x \in_R A$ to indicate that x is chosen uniformly at random from the set A. We use $\{0,1\}^k$ to refer to the binary strings of length k, while $\{0,1\}^*$ is the set of all possible strings. The asterisk here denotes Kleene star. Furthermore we denote by $s_1 \parallel s_2$ the concatenation of the strings s_1 and s_2. When we concatenate group elements $g, h \in G$ like $g \parallel h$ we mean the concatenation of their string representation. Finally, we write $|A|$ to denote the cardinality of a set A.

As a general guideline we use letters in the following way. Points on elliptic curves are indicated by capital letters, while generators of multiplicative groups are often denoted by g and h. Other lower case letters are used for scalar values often from a finite field. Script letters are usually used to denoted classes, sets and machines. Finally, the Greek letters α and β are used for randomly chosen blinding factors.

3.1.1 Algebra

This introduction is very brief as most of the concepts are well known. For some more in depth background most introduction books on algebra will suffice, see for example Dummit and Foote [34]. In this thesis we mostly deal with groups and finite fields. Depending on the context we write a group additively (mostly for elliptic curves) or multiplicatively. The order of a group is its cardinality. The order k of an element P in a group is the smallest integer such that kP is the identity element. The characteristic k of a ring, and hence also of a finite field, is the smallest integer such that adding the multiplicative identity element k times to itself yields the additive identity element.

Finite fields of prime order p and their extension fields of order p^k are indicated by $\mathbb{F}_p$ and $\mathbb{F}_{p^k}$ respectively. Note that the characteristic in both cases is p. The cyclic group generated by an element P is denoted by $\langle P \rangle$. Given a ring R we define R^* to be the group of units. Note that here the asterisk is not the Kleene star. We feel that this overlap is justified because it reflects common practice and because we use the Kleene star only over objects that clearly are sets, such as $\{0, 1\}$.

Finally, we need to introduce some number theory. We write $p|q$ to denote that p divides q, i.e. q is a p-multiple. The group $QR(n)$ is the group of quadratic residues modulo n, i.e. $x \in QR(n)$ if and only if there exists y such that $x = y^2 \pmod{n}$.

3.1.2 Probabilities

Adversaries in cryptographic schemes are almost always considered to be probabilistic. Hence we often have to deal with probabilistics. We write $\Pr[A]$ to denote the probability of an event A, while $\Pr[A \mid B]$ is the conditional probability that A occurs while we know that B is true. Usually the sample space is clear from the context, because it is explicitly mentioned or because of the presence of unbound variables. In a couple of proves however it is beneficial to explicitly mention the sample space. The following notation

$$\Pr_{y \in Y}[A]$$

is used to indicate that the sample space is limited to the set Y.

We use the following definition to indicate events that are very unlikely.

Definition 3.1 (Negligible [42])**.** A function $\mu : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for for any polynomial $f(\cdot)$, there exists N such that for $n \geq N$

$$\mu(n) < \frac{1}{f(n)}.$$

In essence this means that if an event occurs with negligible probability $\mu(k)$ then the expected number of repetitions before this event occurs is $1/\mu(k)$. Since this is super-polynomial in k we can say that they occur so seldom that a polynomial-time algorithm never encounters them.

3.1.3 Interactive Machines

In the previous chapter we saw how Peggy interacted with Victor to show, in zero-knowledge, that she knows the secret passphrase. Interaction is essential for cryptographic protocols and zero-knowledge proofs. Here we give a formal definition of interaction and the participants of such interaction. An interaction usually has a goal to achieve or a problem to solve that is known at the beginning. This is modeled by including this information as common input. Just as in normal conversation, what is said next is based on what happened before as well as randomness. This yields the following definition of a participant.

Definition 3.2 ([7])**.** An *interactive function* $\mathcal{A}$ assigns to each common input $x \in \{0,1\}^*$ and a prefix of a conversation $\eta \in \{0,1\}^*$ a probability distribution on $\{0,1\}^*$. The output $\mathcal{A}_x(\eta)$ is an element chosen at random from this distribution.

Let $\mathcal{P}$ and $\mathcal{V}$ be interactive functions representing a prover and a verifier that interact on a common input x. Furthermore, let α_i (resp. β_i) be the random variable denoting the message send by the prover (resp. verifier) in his i-th move. Then each message is determined by the common input, the previous messages and optionally some built-in randomness:

$$\alpha_i = \mathcal{P}_x(\alpha_1\beta_1 \dots \alpha_{i-1}\beta_{i-1})$$
$$\beta_i = \mathcal{V}_x(\alpha_1\beta_1 \dots \alpha_{i-1}\beta_{i-1}\alpha_i).$$

At the end of the conversation the verifier $\mathcal{V}$ produces a final message that serves as his output. We assume this is indicated by a special symbol. This output will be important in the next section, for now think of it as a verdict on whether the goal has been reached or the problem solved.

Definition 3.3. Let the nth move of the verifier be its last, then we define the *transcript*, $\mathrm{tr}_{\mathcal{P},\mathcal{V}}(x)$, of the interaction between $\mathcal{P}$ and $\mathcal{V}$ on the common input x to be

$$\mathrm{tr}_{\mathcal{P},\mathcal{V}}(x) = \alpha_1\beta_1 \dots \alpha_n\beta_n.$$

where α_i and β_i are as before. Similarly, we define $(\mathcal{P},\mathcal{V})(x)$ to be the random variable that represents the output of the verifier $\mathcal{V}$ after interacting with the prover $\mathcal{P}$ on the common input x.

Note that there is no bound on the computation power of this interactive function. To give such a bound we need a more operation definition of an interactive function. A Turing machine, see Hopcroft et al. [47] for a very nice introduction, is a model that is traditionally used to define complexity. The following definition describes an extended Turing machine that can interact with other such machines and hence models an interactive function.

Definition 3.4. An *interactive (probabilistic) Turing machine [42]* is a Turing machine that models an interactive function. The machine has three additional tapes: one for random bits, one to send messages to another participant and one to receive messages from another participant. An additional mechanism ensures that exactly one machine is active at the same time. See Goldreich [42] for a precise definition.

A participant is said to be polynomially bounded if the number of steps taken by the corresponding Turing machine is polynomially bounded. Since an interactive Turing machine is a model for an interactive function we often write machine instead of interactive function. We note that an algorithm differs from an interactive function in that it only calculates some output given some input. There is no interaction involved.

As we saw in the previous chapter security proofs are often constructed by taking an adversary's program and then use this to construct some useful algorithm. The adversary $\mathcal{A}$ is often called an oracle in this context, hence giving rise to the following definition:

Definition 3.5 (Oracle Machine [7])**.** A interactive Turing machine $\mathcal{K}$ with oracle access to some interactive function $\mathcal{A}$ is an extended interactive Turing machine that has an additional tape for communication with the interactive function $\mathcal{A}$. Then $\mathcal{K}^{\mathcal{A}}(x)$ is a random variable describing the output of $\mathcal{K}$ with oracle $\mathcal{A}$ and input x, where the probability is taken over the random choices of $\mathcal{K}$ and $\mathcal{A}$.

For complexity analysis, only the time needed to write the query and read the answer are considered, the time needed by the oracle to produce the answer is not taken into account.

In the description of zero-knowledge proofs we often want to say that two probability ensembles are indistinguishable. This does not mean that they are exactly the same, it just means that they cannot be distinguished by some predefined method. The following definition defines computational indistinguishability, i.e. the ensembles cannot be distinguished by probabilistic polynomial-time algorithms.

Definition 3.6 (Computationally indistinguishable [42])**.** Let $\mathcal{X} = \{X_w\}_{w \in S}$ and $\mathcal{Y} = \{Y_w\}_{w \in S}$ be two probability ensembles over S. Then $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if for all sufficiently long $w \in S$ and all probabilistic polynomial-time algorithms $\mathcal{D}$

$$|\Pr[\mathcal{D}(X_w, w) = 1] - \Pr[\mathcal{D}(Y_w, w) = 1]|$$

is negligible in the length of w.

3.2 Zero-knowledge proofs of knowledge

The previous chapter we an informal introduction to zero-knowledge proofs. To better understand zero-knowledge proofs we will now discuss them in a more formal setting. We split the discussion of zero-knowledge proofs of knowledge into three parts: interactive proof system, zero-knowledgeness, and proofs of knowledge. We start with interactive proof systems. Interactive proof systems follow the notion of languages from complexity theory. A language is a set of strings or statements that satisfy some requirement. For example, the prime numbers form a language, just as the set of all graphs admitting a Hamiltonian cycle do. Similarly, the traditional notion of a proof induces the language of all statements for which there exists a valid proof. Strings are often called problems because we want to determine whether they are part of the language. How

difficult this is depends on the language. Testing whether a number is a prime is easy, determining whether a statement has a valid proof is hard. Complexity theory gives a classification of these languages according to difficulty.

Informally, a language is in NP if for every element in the language there is a proof, or witness, that can be verified by a polynomial time algorithm. Using the language introduced in the previous section we get the following formal definition.

Definition 3.7 (Complexity class NP)**.** A language L is in NP if there exists a witness-relation $R_L \subset \{0,1\}^* \times \{0,1\}^*$ such that

1. for every element $x \in L$ there exists a polynomially bounded witness, i.e. there exists a polynomial f such that $x \in L$ if and only if there exists a witness y such that $|y| \leq f(|x|)$ and $(x, y) \in R_L$.
2. the validity of a witness y for $w \in L$ can be verified in polynomial time, i.e. there exists a polynomial time Turing machine $\mathcal{M}$ such that $\mathcal{M}(x, y) = 1$ if and only if $(x, y) \in R_L$.

In an interactive proof system the verifier is not given a witness to convince himself of the validity of the statement, instead it is allowed to interact with an all powerful prover to do so. The verifier has to convince himself that the prover does not cheat. The class of NP languages is contained within the class of languages that are provable by an interactive system. To so see this, consider an element from an NP language, then the all-powerful prover can first generate the corresponding witness, and then show this witness to the verifier. The latter can then check the validity of this witness by himself. An interactive proof system is a protocol between a prover and a verifier. This is a good protocol if the prover always convinces the verifier on input from the language, but no prover can, by cheating, (always) erroneously convince the verifier. Formally we have the following definition.

Definition 3.8 (Interactive proof system [43])**.** An *interactive proof system* for a language L is a two-party protocol between a probabilistic polynomial-time verifier $\mathcal{V}$ and a computationally unbounded prover $\mathcal{P}$, satisfying:

- Completeness: For every $x \in L$ the verifier $\mathcal{V}$ always accepts after interacting with the prover $\mathcal{P}$ on common input x.
- Soundness: There exists a polynomial f, such that that for every $x \notin L$ and every prover $\mathcal{P}'$, the verifier $\mathcal{V}$ rejects with probability at least $1/f(|x|)$, after interacting with $\mathcal{P}'$ on common input x.

Such a proof system is written as $(\mathcal{P}, \mathcal{V})$.

By repeating this protocol the error probability can be made arbitrary small.

To define when a proof system is zero-knowledge we first need to formalise the requirement that the verifier "learns nothing substantial" no matter how hard it tries. We say that an verifier learns nothing substantial by behaving adversarially if whatever he learns in this way can also be obtained without interacting with the prover, i.e. by only assuming the validity of the statement itself. Note here the duality with an interactive proof system. There we protected against cheating provers, here we need to protect against cheating

verifiers. If we do not take into account auxiliary information[1] we arrive at the following definition:

Definition 3.9 (Computational zero-knowledge [42])**.** Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system for some language L. We say that $(\mathcal{P}, \mathcal{V})$ is *computational zero-knowledge*, or just *zero-knowledge* on inputs from L if for every probabilistic polynomial time machine $\mathcal{V}'$ there exists a probabilistic polynomial-time algorithm $\mathcal{S}$, called a *simulator*, such that the following two probability ensembles are computationally indistinguishable:

1. $\{(\mathcal{P}, \mathcal{V}')(x)\}_{x \in L}$:= the output of $\mathcal{V}'$ when interacting with $\mathcal{P}$ on common input $x \in L$; and
2. $\{\mathcal{S}(x)\}_{x \in L}$:= the output of $\mathcal{S}$ on input $x \in L$.

An interactive proof system for $\mathcal{S}$ is called zero-knowledge if the prescribed prover $\mathcal{P}$ is zero-knowledge on inputs from $\mathcal{S}$.

In the above definition the simulator $\mathcal{S}$ describes exactly what can be calculated by the validity of the statement alone. An alternative name for this property is simulatability.

An equivalent definition of zero-knowledge can be obtained by looking at the view the verifier has of the interaction instead of its output. By this view we mean the entire sequences of local configurations of the verifier during the interaction. It suffices to only consider the random tape and the messages it receives from the prover as these uniquely determine the sequence of configurations, in fact, this means that the view uniquely determines the transcript we defined earlier. We obtain the following alternative definition.

Definition 3.10 (zero-knowledge, alternative definition [42])**.** Let $(\mathcal{P}, \mathcal{V})$, L and $\mathcal{V}'$ be as in Definition 3.9. We denote by $\text{view}(\mathcal{P}, \mathcal{V}')(x)$ a random variable describing the content of the random tape of $\mathcal{V}'$ and the messages $\mathcal{V}'$ receives from $\mathcal{P}$ during a computation based on common input x. We say that $(\mathcal{P}, \mathcal{V})$ is *zero-knowledge* if for every probabilistic polynomial time interactive machine $\mathcal{V}'$ there exists a probabilistic polynomial-time algorithm $\mathcal{S}$, called a *simulator*, such that the ensembles $\{\text{view}(\mathcal{P}, \mathcal{V}')(x)\}_{x \in L}$ and $\{\mathcal{S}(x)\}_{x \in L}$ are computationally indistinguishable.

These two definitions are in fact equivalent. First of all, Definition 3.10 implies 3.9 since the output of a verifier can be computed in polynomial time from its view. The other direction follows from the fact that for every probabilistic polynomial time $\mathcal{V}'$ there exists a probabilistic polynomial time $\mathcal{V}''$ such that $\text{view}(\mathcal{P}, \mathcal{V}')(x) = (\mathcal{P}, \mathcal{V}'')(x)$.

Finally, we need to define what it means for a machine or a participant to know something. Interaction plays a crucial role. When a participant knows a great secret but never interacts with the outside world then none can learn whether this participant actually knows anything. So, when talking about the knowledge of a prover we actually mean that which can be deduced by interacting with it.

[1] If proofs of knowledge are used as subprotocols we need to take prior knowledge about the preceding transactions into account. In this thesis we will not need this.

It turns out that a good notion of what a prover $\mathcal{P}$ knows about x, as deduced by interaction, can be captured by whatever can be calculated in probabilistic polynomial time on input x and with access to an oracle $\mathcal{P}(x)$, i.e. a prover with as input x. This gives rise to the notion of a knowledge extractor in the following definition. A prover $\mathcal{P}$ knows something if this knowledge can be extracted using an machine that interacts with $\mathcal{P}(x)$ as an oracle.

Definition 3.11 (Proof of knowledge [42])**.** Let L be a language and $R_L \subset \{0,1\}^* \times \{0,1\}^*$ the corresponding binary relation, and $\kappa : \{0,1\}^* \to [0,1]$. Let $\mathcal{V}$ be an interactive probabilistic Turing machine. We say that machine $\mathcal{V}$ is a *knowledge-verifier* for relation R with *knowledge error* κ if the following two conditions hold:

- Non-triviality: There exists an interactive function $\mathcal{P}'$ such that for all $x \in L$, all possible interactions of $\mathcal{V}$ with $\mathcal{P}'$ on common input x are accepting.
- Validity (with error κ): There exists a probabilistic oracle machine $\mathcal{S}$ such that for every interactive function $\mathcal{P}$ and every $x \in L$, machine $\mathcal{K}$ satisfies the following condition:

  > Let $p(x)$ be the probability that $\mathcal{V}$ accepts after interacting with $\mathcal{P}$ on x. If $p(x) > \kappa(x)$ then, on input x and access to oracle $\mathcal{P}_x$, machine $\mathcal{K}$ outputs a string y such that $(x, y) \in R_L$ in expected polynomial time with probability at least $p(x) - \kappa(x)$.

The oracle machine $\mathcal{K}$ is called a *universal knowledge extractor* and κ is called the *knowledge error function*.

The knowledge error function captures the notion that a prover manages to convince the verifier while not actually knowing a witness. The validity requirement then states that if a prover can convince the verifier with a probability higher than the error-function then there is a witness can be extracted with some non-negligible probability.

3.3 Honest-verifier zero-knowledge proofs

General zero-knowledge protocols suffer from two problems: it is very difficult to construct the simulator needed for the zero-knowledge proof and they are not efficient in use. Many repetitions of the protocol are needed to get a low enough soundness error. To counter these problems a weaker variant called honest-verifier zero-knowledge proofs of knowledge was designed. While we in the original protocol needed to deal with every possible verifier (and hence show a simulator for it) we now assume that the verifier will always follow the prescribed protocol defined by the interactive proof system. A verifier that follows this protocol is said to be honest. With this restriction we only need to design a simulator for the view of the honest verifier.

Definition 3.12 (Honest verifier zero-knowledge [42])**.** Let $(\mathcal{P}, \mathcal{V})$ and L be as in Definition 3.10. We say that $(\mathcal{P}, \mathcal{V})$ is *honest-verifier zero-knowledge* if there exists a probabilistic polynomial-time algorithm S such that the ensembles $\{\text{view}(\mathcal{P}, \mathcal{V})(x)\}_{x \in L}$ and $\{S(x)\}_{x \in L}$ are computationally indistinguishable.

Note that there indeed is no universal quantification anymore over all possible verifiers. This makes it much easier to create a simulator, but the notion of zero-knowledgeness is of course much weaker in this setting.

The following example is an honest verifier zero-knowledge protocol by Schnorr [72] that lets Peggy prove knowledge of a discrete logarithm to Victor. This is actually an instance of a Σ-protocol, as we shall see shortly.

Definition 3.13 (Schnorr's proof for discrete logarithms)**.** Let G be a cyclic group of prime order p, written additively, generated by a point $P \in G$. Suppose Peggy generates $x \in_R \mathbb{F}_p$ and sets $X = xP$. Then she can prove to Victor that she knows the discrete logarithm x of X with respect to P by executing the following protocol.

Commitment Peggy generates a random value $r \in_R \mathbb{F}_p$ and sends $R = rP$ to Victor.

Challenge Victor computes a random t-bit challenge e and sends it to Peggy. Here t is such that $2^t < p$.

Response Peggy computes $s = r + xe \pmod{p}$ and sends it to Victor. Victor then verifies that $R = sP - eX$.

One easy trick to analyze protocols of this form is to take $t = 1$, so e is either zero or one. This protocol then reduces to a cut-or-choose type of game. If $e = 0$, then Peggy is asked to reveal the discrete logarithm of R. If $e = 1$ then she has to reveal $r + x$. In neither case she reveals any information, so the protocol seems to be zero-knowledge (in fact it is if t is fixed). Additionally, if Peggy can answer both questions correctly then she knows x as well, so she can cheat with probability at most $1/2$.

Even when t is bigger than one, a cheating Peggy can, having sent R, only correctly answer one challenge. For suppose she can answer two different challenges e, e' correctly, then she can produce s, s' such that $R = sP - eX$ and $R = s'P - e'P$. By subtracting the two equations we find $(s - s')P = (e - e')X$, hence $X = (s - s')/(e - e')P$. Hence, we have recovered the secret key and Peggy knew the secret all along.

Let L be a language with R_L its binary relation as before. We are interested in protocols of the following form, where X is a common input to Peggy ($\mathcal{P}$) and Victor ($\mathcal{V}$), while a witness x such that $(X, x) \in R_L$ is private input to Peggy.

1. Peggy sends a message R.
2. Victor sends a t-bit string e.
3. Peggy sends a response s, and Victor decides to accept or reject based on the data he has seen, i.e. X, R, e, s.

Protocols like this are called Σ-protocols when they satisfy the conditions in the following definition.These conditions ensure that the protocol is a zero-knowledge proof of knowledge with respect to honest-verifiers.

Definition 3.14 (Sigma protocol [30])**.** A protocol is said to be a Σ-protocol for relation R if it satisfies the following requirements.

- The protocol is of the above 3-move form, and it satisfies *completeness*: if $\mathcal{P}, \mathcal{V}$ follow the protocol on input X and private input x to $\mathcal{P}$, with $(X, x) \in R$, then the verifier $\mathcal{V}$ always accepts.

- From any X and a pair $(R, e, s), (R, e', s')$ of accepting conversations on input X, where $e \neq e'$, there exists a polynomial time extractor $\mathcal{K}$ that can extract a witness x such that $(X, x) \in R$. This property is called the *special soundness property.*
- There exists a polynomial-time simulator $\mathcal{S}$, which on input X and a random e outputs an accepting conversation of the form (R, e, s), with the same probability distribution as conversations between the honest $\mathcal{P}, \mathcal{V}$ on input x. This property is called the *special honest-verifier zero-knowledge.*

The relation between completeness and the former definitions of zero-knowledge proofs of knowledge is clear. We note that the special soundness property implies that a cheating prover can only answer one challenge correctly, otherwise the prover would know the secret, so the soundness error is 2^{-t}. Furthermore, the algorithm $\mathcal{K}$ can be used as a knowledge extractor as described in Definition 3.11. Finally we note that for a Σ-protocol the view of the verifier on input x, $\text{view}(\mathcal{P}, \mathcal{V})(x)$ is precisely given by the triple (R, e, s), hence the special honest-verifier zero-knowledge property is a reformulation of the honest-verifier zero-knowledge property from Definition 3.12.

3.3.1 Notation

One of the other advantages of Σ-protocols is that they can be easily combined to give more complicated proofs. For example, to show the knowledge of two discrete logarithms, like a logical and, we can run Schnorr's protocol twice in parallel and even use the same challenge for both runs. A slightly more complicated protocol exists to make a logical or. All of the resulting protocols are again Σ-protocols. This means that we can abstract away from the precise implementation of the proofs and instead focus on what is proved. Camenisch and Stadler [24] introduced a notation to assist in this effort. For example the statement

$$PK\{(\alpha, \beta, \gamma) : y = g^{\alpha} h^{\beta} \wedge n = g^{\alpha} h^{\gamma} \wedge (u \leq \alpha \leq v)\} \tag{3.1}$$

denotes a zero-knowledge proof of knowledge of values α, β and γ such that $y = g^{\alpha} h^{\beta}$ and $n = g^{\alpha} h^{\gamma}$ and $u \leq \alpha \leq v$, where g, h are elements of some group G. Originally, all quantities denoted by Greek letters are only known to the prover, while all other values are known to both the prover as well as the verifier. The reader then has to make the conversion to the actual values that the prover knows, for example that $\alpha = z$. Therefore, if the Greek letter represents a known variable, we will just use this variable. We believe this makes the statements much more readable. When a Greek letter represents something more complicated we leave it as it is. The only downside to this approach is that context determines which values are kept hidden in the proof of knowledge.

3.3.2 Commitments

Commitment schemes [31] are often used in modern complicated cryptographic protocols. They allow a participant in the protocol to choose some value from

a finite set and commit to this choice, i.e. the player cannot change this choice later, without having to reveal which value it chose. Consider the following illustrating example. Peggy wants to commit to some word, so she writes this down on a note, puts the note in a box, and subsequently locks the box while keeping the key. She then gives the locked box to Victor. Clearly, Peggy has now committed to whatever is on the note in the box, she cannot change it anymore. This is called the *binding property*. At the same time, Victor cannot learn what Peggy committed to until Peggy gives him the key. This is called the *hiding property*. A good commitment scheme satisfies both properties.

Commitment schemes are necessary when we want to ensure that parties in a protocol cannot later change their choice. Note that in a protocol we cannot as easily check what the other party does as with the preceding example. One method for obtaining a commitment scheme would be to take any encryption scheme. To commit to a value, first generate a new random public private key pair, and then encrypt the value. This ciphertext together with the public key acts as the commitment. If the encryption scheme is good then this does not reveal anything about the encrypted value, while the plaintext can easily be revealed by providing the private key. We consider two simpler commitment schemes here, starting with the following.

Definition 3.15. Let G be a cyclic group of order p with g as generator. Then the commitment to a value $x \pmod p$ is given by $X = g^x$. This commitment can be opened by revealing x.

Since G is of order p the commitment uniquely determines the committed value. Even with infinite computational power, Peggy cannot change the value she committed to. We say that this scheme is *information theoretically binding*. However, an infinitely powerful adversary could find the committed value. Fortunately, as long as the Discrete Logarithm problem is hard this value cannot easily be recovered, therefore we say that this scheme is *computationally hiding*.

The following commitment scheme is slightly more complicated and has the property that the value that has been committed to cannot be recovered from the commitment.

Definition 3.16 (Pedersen's commitment scheme)**.** Let G be a cyclic group of order p with distinct generators g and h. Then to commit to a value $x \pmod p$ the committer first generates a value $a \pmod p$ and then creates the commit $c = g^x h^a$. The commitment can be opened by revealing both x and a.

Note that for a fixed c and any x there exists an a' such that $c = g^x h^{a'}$. So indeed this commitment does not leak any information, even if the adversary is all powerful. We call this *information theoretically hiding*. It is however consider difficult to find such an a', so the scheme remains binding, but not for an all-powerful committer, hence this scheme is said to be *computationally binding*. This brief introduction to commitment schemes suffices for this thesis. More comprehensive explanations about commitment schemes and their properties can be found in [75, Chap. 24] and [31].

3.3.3 Fiat-Shamir heuristic

The Fiat-Shamir Heuristic [37] is a method for turning any interactive Σ-protocol into a signature scheme. The latter is non-interactive, the proof can be checked without help from the prover, and as a result we lose the simulatability of the proof. The idea is to let the challenge depend on the commitment and a message m by using a hash function $H : \{0,1\}^* \to \mathbb{Z}/(2^t)\mathbb{Z}$, see Definition 2.16. The challenge is given by

$$e = H(R \parallel m).$$

Recall that a cheating prover can correctly answer at most one challenge. So, intuitively, security is guaranteed if it is difficult for the prover to find an input to H that results in this challenge, which is precisely the preimage resistance property of a cryptographic hash-function.

Schnorr's proof of knowledge of a discrete logarithm can be transformed in this way to a signature scheme. The value X now plays the role of the public key while the discrete logarithm x is the private key. By including the message in the challenge for the proof of knowledge of x we obtain a signature scheme as follows.

Definition 3.17 (Schnorr's signature scheme)**.** Let the message space $\mathcal{M}$ equal $\{0,1\}^*$. Then the algorithms are given by:

KeyGen(1^k)**:** The algorithm KeyGen first generates a group G with generator $\mathcal{P}$ of order p with security parameter k. It then generates at random a private key $x \pmod p$ and sets the public key to $X = xP$.

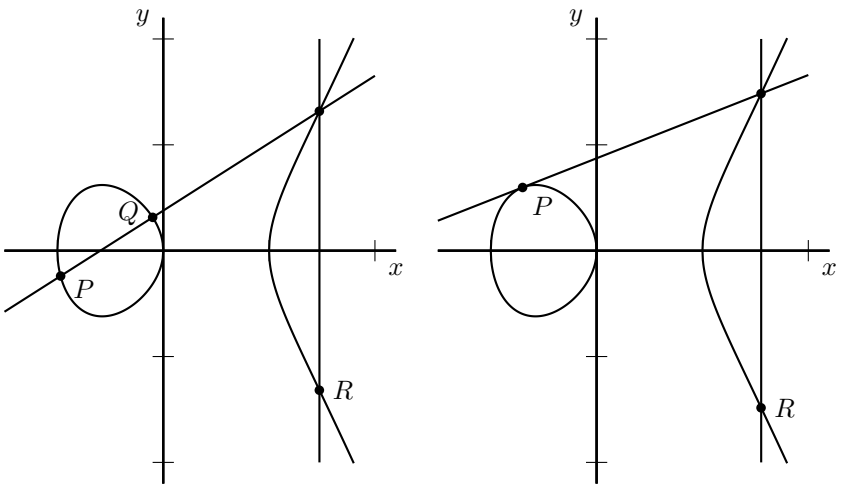
Sign(x, X, m)**:** First the algorithm generates a random number $r \in_R \mathbb{F}_p$, and forms the commitment $R = rP$. Then it calculates the challenge based on the commitment and the message: $e = H(R \parallel m)$. Finally it calculates the response $s = r + xe$. The signature on the message m is then given by $\sigma = (R, e, s)$.

Verify(X, σ, m)**:** The signature σ over m is valid if and only if $e = H(R \parallel m)$ and $R = sP - eX$. The algorithm returns True if it is valid and False otherwise.

We include both R as well as e in the signature here to ease the discussion later on. However, since R can be recovered from s and e, while e can be recovered from R and m only one of them is strictly necessary.

This heuristic can be applied to any Σ-protocol. It is quite useful as it can be used to convert a proof of knowledge of a credential, which can take a more complicated form than the private key in Schnorr's signature scheme, into a signature scheme. This signature then attests that the signer knows the values that satisfy the statement in the proof of knowledge. With Schnorr's scheme this lead to a proper signature scheme because the signer knows the private key. We will encounter some protocols that are modeled around this principle. In order to simplify exposition we use the notation $SPK\{(\ldots) : \text{statement}\}(m)$ to denote a signature on a message m derived from a proof of knowledge about the statement. For example the Schnorr signature from the previous definition can then be written as

$$\sigma = SPK\{(x) : X = xP\}(m).$$



(a) Addition: $P + Q = R$ (b) Addition: $P + P = R$

Figure 3.1: Example of using the geometric chord-tangent rule for adding points on the elliptic curve $y^2 = x^3 - x$ over the real numbers

3.4 Elliptic curve cryptography

In this section we will give a very brief introduction to elliptic curves. The goal of this section is twofold. The first is to show how elliptic curves give rise to an abelian group structure. Given this group structure we can use elliptic curves in schemes based on the hardness of the discrete logarithm problem. The second goal is to give a precise enough definition of elliptic curves to be able to define pairings, an important tool in many protocols. Most of the information presented in this section is well known and can be found in any textbook on elliptic curves, see for example Silverman [73].

3.4.1 Elliptic curves

An elliptic curve E over a field K is defined by a non-singular Weierstrass equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{3.2}$$

where $a_1, a_2, a_3, a_4, a_6 \in K$. The set $E(K)$ of points on such a curve is the set of points $(x, y) \in K \times K$ satisfying equation (3.2) together with a point O, which we call the point at infinity. If the characteristic p of the field K is not 2 or 3 then a linear transformation can be used to bring equation (3.2) into a simpler form

$$y^2 = x^3 + ax + b, \tag{3.3}$$

where $a, b \in K$ and $4a^3 + 27b^2 \neq 0$. The latter equation ensures that the cubic does not have roots with multiplicity more than 1.

The points $E(K)$ form an abelian group under the chord-and-tangent rule for adding two points. Traditionally this group is written additively. An example of this rule for the curve $y^2 = x^3 - x$ over the real numbers is shown in Figure 3.1. The negative of a point $P = (x, y_1)$ is given by $-P = (x, y_2)$ where y_1, y_2 are

the two roots defining equation (3.3). To add two distinct points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ draw a line through them and find the third intersection R' with the curve. The sum of P and Q is now obtained by mirroring R' in the x-axis, so $P + Q = -R'$. See Figure 3.1a. If $P = Q$ we draw the tangent instead, see Figure 3.1b. The point at infinity, O, acts as the identity element. For k a positive integer we write kP to denote P added k times to itself.

The geometric construction gives rise to a set of rational formulas for the addition of points. If $P = (x_P, y_p)$ and $Q = (x_Q, y_Q)$ are both points on the curve $E(K)$ then $P + Q = R = (x_R, y_R)$ is given by

$$\begin{aligned} x_R &= \lambda^2 - x_P - x_Q, \\ y_R &= \lambda(x_P - x_R) - y_P, \end{aligned}$$

where

$$\lambda = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{if } P \neq Q, \\ \frac{3x_P^2 + a}{2y_P} & \text{if } P = Q. \end{cases}$$

For fields of characteristic 2 and 3 similar addition and doubling formulas can be found.

In the remainder of this thesis we will only use elliptic curves over a finite field $\mathbb{F}_q$ of order q and characteristic p. In this setting $E(\mathbb{F}_q)$ is a finite abelian group of the following form: $E(\mathbb{F}_q) \cong C_{n_1} \oplus C_{n_2}$, where C_n is a finite group of order n, and n_2 divides both n_1 and $q - 1$. Hasse's theorem states that $\#E(\mathbb{F}_q) = q + 1 - t$ for $|t| \leq 2\sqrt{q}$.

Definition 3.18. An elliptic curve over a finite field $\mathbb{F}_q$ of order q and characteristic p and cardinality $\#E(\mathbb{F}_q) = q + 1 - t$ is called *supersingular* if $p|t$ and *non-supersingular* or *ordinary* otherwise.

Example 3.19 (An elliptic curve over $\mathbb{F}_{11}$)**.** Consider the curve $E(\mathbb{F}_{11})$ defined by the equation $y^2 = x^3 + 5x + 1$. The point $P = (0, 1)$ is clearly on the curve and it turns out to be a generator of the group. By repeatedly adding P to itself we find:

$$\begin{aligned} P &= (0,1), & 2P &= (9,4), & 3P &= (7,4), & 4P &= (8,5), \\ 5P &= (6,7), & 6P &= (6,4), & 7P &= (8,6), & 8P &= (7,7), \\ 9P &= (9,7), & 10P &= (0,10), & 11P &= O. \end{aligned}$$

The curve contains no other points, hence we have that $t = -1$, since $q = p = 11$ and thus the curve is ordinary.

3.4.2 Elliptic curve cryptography

The use of elliptic curves for cryptography was first proposed by Miller [59] and Koblitz [54] around 1986. Until then, most cryptosystems were based on the hardness of the discrete logarithm problem used multiplicative (sub)groups of finite fields. Elliptic curves were presented as alternative to these multiplicative groups. We can state the elliptic curve discrete logarithm problem as follows:

Problem 3.20 (Elliptic Curve Discrete Logarithm Problem (ECDLP))**.** Let $P \in E(\mathbb{F}_q)$ be a point of order n with $\gcd(n, q) = 1$. Then the elliptic curve discrete logarithm problem in $\langle P \rangle$ is the following: given $Q \in \langle P \rangle$ find the integer $k \pmod{n}$ such that $Q = kP$.

In the following we will simply refer to ECDLP by DLP, and no longer differentiate between the underlying groups.

One might wonder what the advantage is of using elliptic curves as opposed to the much simpler finite fields. The reason is that solving the discrete logarithm problem for an elliptic curve group is much harder than solving it for a finite field. In fact, for a finite field discrete logarithm setting with a 128 bit security level, ECRYPT currently advises to use a generator from a 3248-bit group with order of 256 bits [76]. This means that every group element takes 3248 bits to represent. For elliptic curves on the other hand the group elements can be represented by approximately 256 bits.

Let n be the order of a point P on the elliptic curve. The best known generic algorithm for solving ECDLP is Pollard's rho method [69]. This method has expected runtime of $O(\sqrt{n})$. A result by Pohlig and Hellman [67] shows that it is sufficient to solve the problem with respect to the prime factors of n. Hence n must contain a large prime factor to obtain optimal security. For discrete logarithm problem in finite fields index calculus algorithms exists that in some cases achieve a complexity of $O(\sqrt[3]{n})$ [53], hence explaining the differences in bit length.

For special types of elliptic curves, most notably the super-singular ones, we can improve upon the Pollard rho method by using Weil or Tate pairings. In Section 3.5 we will examine these pairings in more detail and also see how we can benefit from them. Using these Weil or Tate pairings we can embed the group $E(\mathbb{F}_q)$ in $\mathbb{F}_{q^k}^*$ for some integer k. Thus, the ECDLP in $E(\mathbb{F}_q)$ can be reduced to solving the DLP in $\mathbb{F}_{q^k}^*$. Let $E(\mathbb{F}_q)$ be a cyclic group of order n then a necessary condition on k is that n divides $q^k - 1$, it turns out that this condition is also sufficient for a pairing to exist. It therefore makes sense to define the empbedding degree.

Definition 3.21 (Embedding Degree)**.** Let $E(\mathbb{F}_q)$ be an elliptic curve of order n. Suppose that $\gcd(n, q) = 1$, then the *embedding degree* of $E(\mathbb{F}_q)$ is the smallest positive integer k such that $n | q^k - 1$.

If the embedding degree is small then using index calculus methods in the field $\mathbb{F}_{q^k}^*$ may give a faster method for solving the ECDLP. It is known that supersingular curves have embedding degree at most 6, so they are especially vulnerable to this attack, which is sometimes called the MOV-attack after the authors Menezes, Okamoto and Vanstone [57] that first described it. It is, however, quite rare that a randomly chosen elliptic curve has a small embedding degree.

Let us briefly consider two protocols that use elliptic curve cryptography. The first is the Diffie-Hellman key-exchange protocol [33]. Suppose Alice and Bob want to create a secret key, known only to them, over an insecure channel without having any previous contact. Let $E(\mathbb{F}_q)$ be an elliptic curve over $\mathbb{F}_q$

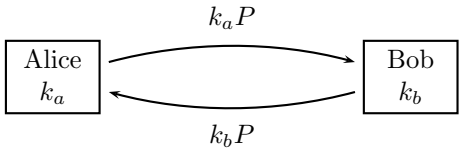


Figure 3.2: Two-party one-round key exchange protocol.

and let $P \in E$ be a publicly known point of order n. Alice generates a random integer $k_A \pmod n$ and sends k_AP to Bob. Likewise, Bob chooses a random integer $k_B \pmod n$ and sends k_BP to Alice. See Figure 3.2. The common key is now given by $Q = k_Ak_BP$. Alice finds Q by multiplying the value she received from Bob by her random integer k_A. Bob does likewise. The task of an attacker is to find $Q = k_Ak_BP$ given only k_AP, k_BP and the public point P. This is the elliptic curve variant of the Computational Diffie-Hellman problem.

Problem 3.22 (Computational Diffie-Hellman (CDH) Problem)**.** Let G be a cyclic group with generator P of order p. Then the Computational Diffie-Hellman problem is to calculate, given $X = xP, Y = yP$ the point $Z = xyP$.

Another example is ElGamal encryption [35], which we briefly restate here to show how things change when moving to elliptic curves. Let $E(\mathbb{F}_q)$ be an elliptic curve and $P \in E$ a point on that curve as above. Suppose Alice wants to encrypt a message $M \in E$ and send it to Bob. She knows Bob's public key $X_B = x_BP$, while Bob also knows the corresponding private key x_B. Alice first generates a random integer $k \pmod n$. She then sends the pair $(kP, M + kX_B)$ to Bob. Note that this algorithm is probabilistic. Bob receives the tuple (K, S). To recover the message he first calculates x_BK and then subtracts this from S. No adversary can win the IND-CPA game from the previous chapter provided the computational Diffie-Hellman problem is hard.

3.5 Pairings

After the discovery of the MOV-attack supersingular curves were considered to be a security risk. This attitude remained until Joux [50] demonstrated the first benign application of pairings in 2000. Since then a whole range of new protocols have been designed making use of the very property of supersingular curves that was until then considered a weakness: the existence of pairings. In this section we will mainly study pairings as a black-box operation on elliptic curves. We only briefly describe the inner workings of the Weil and Tate pairings. A more thorough and technical description of the theory behind pairings can be found in survey papers by Galbraith [40], Vercauteren [78] and an early paper by Joux [51]. The survey papers by Menezes [56] and Paterson [65] give a more general introduction to protocols using pairings.

Definition 3.23 (Pairings)**.** Let G_1 and G_2 be cyclic groups of order p written additively, and G_3 an cyclic group of of order p that is written multiplicatively. Then a *pairing* is a function

$$e : G_1 \times G_2 \to G_3$$

satisfying the following properties.

Bilinearity: For all $P, P' \in G_1$ and $Q, Q' \in G_2$ we have

$$e(P + P', Q) = e(P, Q)e(P', Q)$$

and

$$e(P, Q + Q') = e(P, Q)e(P, Q').$$

Non-degeneracy:

- For all $P \in G_1, P \neq O$, there exists $Q \in G_2$ such that $e(P, Q) \neq 1$.
- For all $Q \in G_2, Q \neq O$, there exists $P \in G_1$ such that $e(P, Q) \neq 1$.

Efficiency: The function e can be computed efficiently.

The pair (G_1, G_2) is called a bilinear group-pair if a pairing exists satisfying the above properties. We define $\mathcal{I}(1^k)$ to be a pairing instance generator. It generates groups G_1, G_2 and G_3 of size approximately 2^k and a pairing $e : G_1 \times G_2 \to G_3$.

We note that in the literature the groups G_1 and G_2 are sometimes written multiplicatively instead. However, since in all implementations G_1 and G_2 are (subgroups of) elliptic curves, we use the additive notation. Also, the bilinearity of the pairing explains why they are sometimes called *bilinear maps*. In this thesis we will mostly use the term pairings instead. As a consequence of the bilinearity we get the following lemma:

Lemma 3.24. *Let e be a pairing as above. Let $P \in G_1$, $Q \in G_2$ and $a, b \in \mathbb{F}_p$ then the following statements hold:*

1. $e(P, O) = e(O, Q) = 1$
2. $e(-P, Q) = e(P, Q)^{-1}$
3. $e(aP, bQ) = e(P, Q)^{ab}$

In the definition G_1 and G_2 are named seperately. This makes sense because in the underlying Weil and Tate pairing these groups are indeed different. However, many protocols using pairings require that $G_1 = G_2$. This turns out to be possible if the underlying elliptic curve is supersingular. Other protocols require that there exists an efficiently computable homomorphism from G_2 to G_1. Note that since both G_1 and G_2 are cyclic groups of the same order in theory such homomorphism always exists. However, for some configurations calculating this homomorphism is as hard as finding discrete logarithms. Following Galbraith et al. [41] we define the following three types of pairings.

Type 1: $G_1 = G_2$;
Type 2: $G_1 \neq G_2$ and there is an efficiently computable homomorphism $\phi : G_2 \to G_1$; and
Type 3: $G_1 \neq G_2$ and there are no efficiently computable homomorphisms between G_1 and G_2.

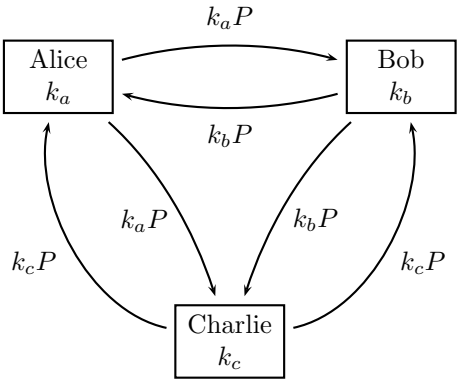


Figure 3.3: Three-party one-round key exchange protocol.

It is the last property of Lemma 3.24 that is most often used in cryptographic protocols. In the previous section we saw how elliptic curves could be used to make a two-party one-round key agreement protocol. For a long time it has been an open question whether a three-party one-round key agreement protocol was possible. In 2000 Joux showed that by using pairings this was indeed the case [50]. At the beginning of the protocol Alice, Bob and Charlie choose random secret integers $k_a, k_b, k_c \pmod p$ respectively. Alice then broadcasts k_aP to the other to the other two parties. Bob and Charlie similarly broadcast k_bP and k_cP. After receiving the broadcasts Alice (and also Bob and Charlie by symmetry) can compute the shared key $K = e(k_bP, k_cP)^{k_a} = e(P,P)^{k_ak_bk_c}$, see Figure 3.3. Here we need to use a Type 1 pairing since we use that $G_1 = G_2$. The security of this protocol is based on the assumption that the Bilinear Diffie-Hellman problem is hard.

Problem 3.25 (Bilinear Diffie-Hellman Problem (BDHP))**.** Let e be a Type 1 pairing from $G_1 \times G_1$ to G_T. Then the *bilinear Diffie-Hellman Problem (BDHP)* is the following: given P, aP, bP, cP compute $e(P,P)^{abc}$.

Another way of looking at a pairing is in terms of the following problem.

Problem 3.26 (Cross-group Diffie-Hellman Problem)**.** Let G_1, G_2 be groups of prime order p. Then the *Cross-group Diffie-Hellman problem* is to decide, given $P, A = aP \in G_1$ and $Q, B = bQ \in G_2$, whether $a = b$.

This problem can easily be solved if there exists a pairing from $G_1 \times G_2$ to G_T. The values a and b are equal whenever the following equation holds:

$$e(A,Q) = e(P,B).$$

Often, protocols involving pairings rely primarily on the pairings ability to solve the Cross-group Diffie-Hellman problem. We note that if $G_1 = G_2$ then the cross-group Diffie-Hellman problem reduces to the ordinary decisional Diffie-Hellman problem. Consequently, if a type I pairing exists for G_1 then the DDH problem is trivially solvable as well.

Boneh, Lynn and Shacham [16] used pairings to create a signature scheme in which signatures are roughly half the length of traditional ElGamal-like signature schemes. This scheme is more thorougly covered in the next section. Yet another problem that is insolvable without pairings is identity-based encryption. Boneh and Franklin proposed the solution [15]. Roughly speaking this scheme allows the use of the identity of a person to act as a public key, hence preventing problems with the web of trust that plagues ordinary public key encryption.

3.6 Signature schemes

In Section 2.3 we introduced signatures as the digital counterpart to the traditional handwritten signatures. In the following chapters we describe protocols that often make use of specific signature schemes. We describe two schemes here. The first scheme was proposed by Chaum and Pedersen [28] and later formalized by Boneh, Lynn and Shacham [16]. This scheme is at the basis of the first self-blindable credential protocols without revocation [5, 79]. The second scheme, proposed by Boneh and Boyen, is used by the self-blindable credential protocols with revocation [36, 52] as well as by our protocol.

3.6.1 Chaum and Pedersen signatures

In their 1992 paper Chaum and Pedersen [28] describe a very simple signature scheme that can be used in an early form of credentials. To obtain a signature on a message $M \in G$ the latter is multiplied with the private key x of the signer. Intuitively, it is hard to forge such a signature from only $X = xP$ and M as that would involve solving the assumed to be hard Computational Diffie-Hellman problem.

To verify a signature it is sufficient to ensure that the quadruple $(P, X = xP, M, \sigma = xM)$ is a Diffie-Hellman tuple, i.e. it is of the form (P, aP, bP, abP) for some values of P, a, b. In the protocols by Chaum and Pedersen it is essential that the prover participates in the verification. Therefore it makes sense to use an interactive zero-knowledge proof between the signer and the verifier where the former proves that the discrete logarithm of σ with respect to M and X with respect to P are the same:

$$PK\{(x) : \sigma = xM \wedge X = xP\}.$$

For ordinary signatures one would prefer a verification protocol that can be executed without help from the signer. This could be achieved by converting the interactive proof into a non-interactive one via the Fiat-Shamir heuristic.

The Chaum and Pedersen signatures do suffer from one notable defficiency: given a signature on M it is trivial to make signatures on yM for any $y \in \mathbb{Z}_p$ simply by multiplying the original signature by y. However, it is precisely this property that makes them very suitable for self-blindable credentials, see Section 6.1.

Boneh, Lynn and Shacham [16] made this protocol more general by allowing any message in $m \in \{0,1\}^*$ and then hashing them onto the group to obtain $M = H(m)$. This prevents chosen message attacks and allows for a security

proof in the random oracle model, see Section 3.7. Second, they replaced the proof of knowledge by a pairing operation. Now the signature $\sigma = xM$ can be checked without help from the signer. The following definition gives the complete protocol.

Definition 3.27 (BLS signature [16])**.** The Boneh, Lynn and Shacham (BLS) signature scheme [16] is a signature scheme with G a cyclic group of order p, $P \in G$ a generator and $H : \{0,1\}^* \to G$ a hash function mapping onto the group. The signature scheme consists of the following three algorithms:

KeyGen(1^k)**:** Generate a cyclic group G of order p, where p is k-bits long. Then pick a generator P of G at random, pick a random secret key $a \in \mathbb{Z}_p$, and set the corresponding public key $A = aP$. It returns descriptions of the group G, the generator P, and the hash function $H : \{0,1\}^* \to G$ in addition to the public key A and private key a.

Sign(a, m)**:** Given a secret key a, and a message $m \in \{0,1\}^*$, compute $M = H(m)$ and $\sigma = aM$. The signature is $\sigma \in G$.

Verify(A, σ, m)**:** Given a message $m \in \{0,1\}^*$, a signature $\sigma \in G$ and a public key A, compute $M = H(m)$ and verify that (P, A, M, σ) is a valid Diffie-Hellman tuple.

In practise the algorithm will be used in the bilinear maps setting with (G_1, G_2) a bilinear group pair and $e : G_1 \times G_2 \to G_T$ a bilinear map. In this setting $P \in G_1$ and $H : \{0,1\}^* \to G_2$ maps onto G_2. The verification check can then be replaced by the following check:

$$e(P, \sigma) = e(A, M).$$

The use of a hash function that hashes onto the curves does restrict the possible bilinear group pairs that can be used [41].

3.6.2 Boneh and Boyen signature

The BLS scheme from the previous section has several disadvantages. First, the security of the scheme can only be shown in the random oracle model. This model is not universally accepted, see Section 3.7 for more information. Furthermore, the scheme is not randomized. Finally, the full domain hash that maps onto the elliptic curve is tricky to construct and analyze [16, 41]. The scheme proposed by Boneh and Boyen [13] tackles these problems by assuming the Strong Diffie-Hellman (SDH) problem is hard. Strong here means that we can derive better results from this assumption, however this also means that the SDH problem might be easier to solve in practice than for example the computational Diffie-Hellman problem. The advantage of assuming that the SDH problem is hard is that the random oracle model is no longer needed to show resistance against chosen message attacks.

Let (G_1, G_2) be a bilinear group pair of prime order p as before. Let P be a generator of G_1 and Q a generator of G_2. Then the Strong Diffie-Hellman problem is defined as follows.

Problem 3.28 (q-Strong Diffie-Hellman (SDH) problem [13])**.** The q-Strong Diffie-Hellman problem defined over a bilinear group pair (G_1, G_2) is defined

as follows: given as input a $(q+3)$ tuple $(Q, xQ, x^2Q, \ldots, x^qQ, P, xP)$, where $P \in G_1$ and $Q \in G_2$, output a SDH-pair $(\frac{1}{x+k}Q, k)$, where $k \in \mathbb{Z}_p$ is chosen by the adversary. An algorithm $\mathcal{A}$ has advantage ϵ in solving the q-SDH problem in (G_1, G_2) if

$$Pr\left[\mathcal{A}(Q, xQ, \ldots, x^qQ, P, xP) = \left(\frac{1}{x+k}Q, k\right)\right] \geq \epsilon,$$

where the probability is taken over the random choice of the generators P and Q, the key $x \in \mathbb{Z}_p$ and the random bits of $\mathcal{A}$.

Definition 3.29. We say that the (q, t, ϵ)-SDH assumption holds in (G_1, G_2) if no t-time algorithm has an advantage at least ϵ in solving the q-SDH problem in (G_1, G_2).

The full scheme proposed by Boneh and Boyen is described as follows.

Definition 3.30 (Boneh and Boyen signature)**.** Let (G_1, G_2) be a bilinear group pair and $P \in G_1, Q \in G_2$ generators for the respective groups. Let p be the order of the cyclic groups. The signature scheme consists of the following three algorithms:

KeyGen(1^k)**:** Run $\mathcal{I}(1^k)$ to obtain the groups G_1, G_2 and the pairing e. Choose two private keys $a, b \in_R \mathbb{Z}_p^*$ and set the public keys $A = aP, B = bP$ accordingly.

Sign$((a, b), m)$**:** Given a private key (a, b) and a message m choose a random value $r \in \mathbb{Z}_p \setminus \left\{-\frac{a+m}{b}\right\}$ and compute $C = \frac{1}{m+a+rb}Q$. The inverse is calculated modulo p. The signature is (C, r).

Verify$((A, B), \sigma, m)$**:** Given a public key (A, B) and a signature $\sigma = (C, r)$ on a message m verify that $e(mP + A + rB, C) = e(P, Q)$.

A simpler version of this scheme is obtained by fixing $r = 0$. Note this means we have to ensure that $m \neq -a$. We shall use the name Simple Boneh and Boyen signatures to indicate this when confusion could arise.

This scheme can resist adaptive-chosen message attacks provided the SDH-assumption holds. We only give a sketch of the proof here. An almost complete proof can be found in Theorem 7.10. First we prove that the simple scheme can withstand weak chosen message attacks. In this game the attacker has to declare all the messages it will query before it even gets the public key. Suppose we have such a succesfull attacker then we use it to solve an SDH-instance. We first extract the signatures the attacker will request from the SDH-instance. Then we run the attacker and provide the signatures. Finally the attacker outputs a forged signature. From this signature we recover a solution to the SDH-instance.

The full scheme can handle an adaptive attacker because of the extra freedom we have in choosing r. Suppose we have an attacker than can successfully make adaptive chosen ciphertext attacks. We proceed as before, but now generate signatures on random messages $m_i \in \mathbb{Z}_p$. Whenever the adversary asks for a signature on a message m we pick r such that $m + br = m_i$ for a new i and return the corresponding signature. This trick allows us to fake signatures

even though we did not know on which message they were suppose to be at the start.

While it was easy to generate signatures on derived private keys for the Chaum Pedersen signatures this is not possible for the Boneh and Boyen signature scheme since even under adaptive attacks an adverary cannot make existential forgeries. This makes these types of signatures more suitable for self-blindable credentials with revocation, as we shall see. The scheme by Emura et al. [36] uses the simple version for their self-blindable credential scheme, see Section 6.2. Our protocol uses the full version to ensure unlinkability, see Chapter 7.

3.7 Random oracle model

In section 2.2.1 we saw that the proofs of security for cryptographic protocols should be seen as relativistic arguments. If we can break said protocol, then we can also succeed in breaking an underlying problem which we assume to be hard. Stronger intractability assumptions allow more things to be proven. The signature scheme by Boneh and Boyen [12] described in Section 3.6.2 is an example where the Strong Diffie-Hellman assumption allows us to prove that the scheme is secure under adaptive chosen-ciphertext attacks. The random oracle model is another way of introducing a reasonable, but strong, assumption to facilitate security proofs.

Bellare and Rogaway [8] gave the first formal description of the random oracle model in 1993. The main ideas, however, were already present in other, earlier work, for example by Fiat and Shamir [37]. Many protocols use hash-functions. In practice these are implemented using for example MD5 or SHA-1. In the random oracle we make the assumption that these hash-functions are in fact truly random functions, so we can replace them by an idealized hash-function that on input of a new query picks, uniformly at random, a value from its codomain. If this query was asked before it will return the previously chosen value. Such a function is called a *random oracle*. The hash functions used in practice are not truly random, because they are specified by a short procedural description and hence exhibit structure. It is precisely this structure that we assume not to be exploitable/noticeable in the hash functions we use in practise. Proofs in the random oracle model follow the following steps.

1. Define an efficient protocol for your problem in an ideal model where all parties, including any adversaries, share a random oracle R.
2. Prove the security of your protocol in this ideal model, using the random oracle R.
3. Replace all calls to the random oracle R by a short hash-function.

The final step is only heuristic. The trust is based on experience rather than on a proof. Note that, in the ideal model, the attacker cannot use the specific structure of the hash-function as it is replaced by a random oracle. Thus the proof of security that is obtained in this way shows that if there is a successful attacker for the system then it must use the structure of the specific hash-function.

Most security games, see section 2.2.1, allow an active adversary, so for example an attacker can make signature queries or decryption queries. If we want to

make a reduction to the underlying hard problem we should be able to answer these queries. In the proof for the Boneh and Boyen signatures we could use the SDH-instance to create valid answers to these queries. The following theorem shows an example of how the random oracle model can be used to prove that a signature scheme is secure against an adaptive chosen message attack. The active queries are satisfied by fixing the response of the random oracle in such a way that we can generate signatures over them. The signature scheme is based on the RSA-signature scheme in Definition 2.15. Instead of allowing only messages modulo n, a hash-function H is used to map any string into the ring $\mathbb{Z}/n\mathbb{Z}$. Since this hash-function maps into the full domain $\mathbb{Z}/n\mathbb{Z}$ of the signature scheme it is sometimes called a full-domain hash-function.

Theorem 3.31. *Let $n = pq$ be an RSA-modulus and $(\mathcal{P}, \mathcal{S}) = (e, d)$ be the public-private key pair and let $H : \{0,1\}^* \to \mathbb{Z}/n\mathbb{Z}$ be a full-domain cryptographic hash-function. Then the RSA-FDH signature scheme is given the following two algorithms:*

Sign(d, m)**:** *The signature σ on message m with private key d is given by $\sigma = H(m)^d \pmod n$.*

Verify(e, m, σ)**:** *A signature σ on a message m with public key e is valid if and only if $\sigma^e = H(m) \pmod n$.*

There is no adversary that can with advantage ϵ produce an existential forgery on the RSA-FDH signature scheme using an adaptive-chosen message attack making k signature queries provided that the $((\epsilon - 1/n)/k)$-RSA assumption holds, i.e. no adversary has advantage $((\epsilon - 1/n)/k)$ in solving the RSA problem.

In the subsequent proof we see a technique that is very typical for proofs in the random oracle model. As usual we build an algorithm $\mathcal{B}$ that uses a forger to solve some difficult problem, in this case RSA. Now, the attacker not only makes signature queries that $\mathcal{B}$ should answer, but it also makes hash-queries that $\mathcal{B}$ is responsible for answering. So our algorithm acts as the random oracle. This is precisely what allows us to cheat.

Furthermore, we can see in the theorem the effect of the probabilistic nature of the adversary. In order to show that no adversary has advantage ϵ in creating an existential forgery we show that if such an adversary exists we can solve the RSA problem with probability at least $((\epsilon - 1/n)/k)$. Note that this means our reduction will not always succeed. We see why in the following proof.

Proof. This proof is based on the proof in Bellare and Rogaway [8]. Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm that creates an existential forgery on the RSA-FDH signature scheme with probability at least ϵ. We build an algorithm $\mathcal{B}$ that breaks the RSA-assumption. First $\mathcal{B}$ runs the RSA-instance generator and obtains an RSA-modules n, the public key e and a value $y = x^e \pmod n$. The goal of $\mathcal{B}$ is to efficiently find the value x. Algorithm $\mathcal{B}$ first chooses a value $t \in \{1, \ldots, k\}$ at random. We assume that if $\mathcal{A}$ makes a signing query on a message m then it has already queried $H(m)$, this is without loss of generality as we can always make the query ourselves. Furthermore we assume, again w.l.o.g., that all the queries are unique. Algorithm $\mathcal{B}$ replies to queries as follows:

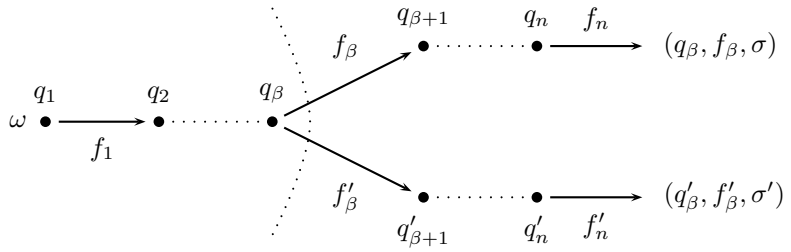


Figure 3.4: A fork in the responses f_i to the queries q_i at position β. If the fork is successful we obtain two different signatures σ and σ' at the end.

- Let m_i be the i-th hash-query that $\mathcal{A}$ makes to the random oracle. If $i = t$ then $\mathcal{B}$ answers by returning y. Otherwise it picks a value $r_i \in_R \mathbb{Z}_N^*$ at random and returns $y_i = r_i^e \pmod N$.
- Suppose $\mathcal{A}$ makes a signing query on m. If $y = m_t$ then $\mathcal{B}$ aborts and admits failure. Otherwise, it replies with r_i for i such that $m = m_i$.

Eventually $\mathcal{A}$ outputs a forged signature σ over a message m. If $m \neq m_t$ then $\mathcal{B}$ aborts and admits failure. We know $\sigma^e = H(m)$ since the signature is valid. If $m = m_t$ and $\mathcal{A}$ queried the random oracle for m then $\sigma^e = y \pmod N$. So, we have found the desired solution to the RSA-problem. Note that the signing oracle was not queried for m, since then $\mathcal{B}$ would have aborted.

With probability $1/N$ the adversary $\mathcal{A}$ guessed the hash-value all by itself. So with probability $(\epsilon - 1/N)$ adversary $\mathcal{A}$ did indeed query $\mathcal{B}$ for the hash of m. Now t is chosen at random and its value is independen of the view of $\mathcal{A}$, so $\mathcal{B}$ succeeds with probability $(\epsilon - 1/N)/n$ as desired. □

3.7.1 Forking Lemma

Some proofs in the random oracle model require a bit more sophistication. Recall the Schnorr-signatures described in Section 3.3.3. For a public-private keypair $(\mathcal{P}, \mathcal{S}) = (X = xP, x)$ a signature on a message m is a tuple (R, e, s), where $R = rP$, for a randomly chosen $r \in \mathbb{Z}_p$, e is the hash of m and R, so $e = H(R \parallel m)$, and $s = r + ex$. Such a signature is verified by checking that for $h = H(R \parallel m)$ the equation $R = sP - eX$ holds. This time setting some of the answers of the random oracle to a predetermined value will not help in making a useful reduction to some hard problem such as the Discrete Logarithm Problem. A more sophisticated treatment is needed.

For Schnorr's proof of knowledge of a discrete logarithm protocol it is easy to show that the prover must know the secret key because of the special soundness property. The trick is to run the prover twice, with the same randomness but to provide it with a different challenge. From the resulting transcripts (R, e, s) and (R, e', s') the secret key can be recovered. With Schnorr-signatures this replay trick does not work as the challenge is now determined by the hash-function and hence fixed. This is where the random oracle model shows its use: we can run the prover twice, but with a different random oracle. If we manage to change the hash-query that results in the challenge h then we can obtain two signatures (R, e, s) and (R, e', s') and recover the secret key as before. Unfortunately, this heuristic argument is too simplistic. The adversary can query the oracle more

than once and make choices depending on all the previous answers (for example because it first searches for a suitable response). Fortunately, we can see, after the first run, which query was for $R \parallel m$ and hence should be altered. However, this changes the run of the adversary from that point forward. For the original answer we know the adversary ran succeeded in producing a forgery, for the new answer this is not a priori known. Maybe our run to completion had a very low probability and consequently the adversary will almost never run to completion on the new hash-response. It turns out this is not true. Pointcheval and Stern [68] gave a formal proof that this splitting trick will indeed succeed with non-negligible probability.

Before we can give the actual proof we first need to prove the so-called heavy rows lemma. Essential in the proof of the forking lemma is that we want to show that given a succesful forgery and corresponding random oracle, we can modify the random oracle and still obtain a succesful forgery with reasonable probability. In the lemma below you can think of the rows X as being all possible executions, determined by the randomness and the choice of the random oracle, up to the critical hash query, while the columns Y represent all possible continuations for the remaining hash queries. The set A consists of those combinations that result in a forgery. When we make a forking attack on a forger we change the responses of the random oracle from the critical query onwards, so we only change the column. Therefore, it makes sense to investigate rows where the probability of succes is high. The lemma bounds the probability that a succesful run is in such a row.

Lemma 3.32 (Heavy rows lemma)**.** *Let $A \subset X \times Y$ such that $\Pr[(x,y) \in A] \geq \eta$. For any $\alpha < \eta$ define the set of heavy rows as:*

$$B = \left\{ (x,y) \in X \times Y \mid \Pr_{y' \in Y}[(x,y') \in A] \geq \eta - \alpha \right\},$$

then the following statements hold:

(i) $\Pr[B] \geq \alpha$

(ii) $\Pr[B \mid A] \geq \alpha/\eta$.

Proof. (i) Assume, to obtain a contradiction, that $\Pr[B] < \alpha$. Let $\bar{B} = (X \times Y) \setminus B$ be the complement of B. From the definition of B it follows that

$$\forall (x,y) \in B \quad \Pr_{y' \in Y}[(x,y') \in A] \geq \eta - \alpha, \tag{3.4}$$

so we have that $\Pr_{y' \in Y}[(x,y') \in A] < \eta - \alpha$ for all $(x,y) \in \bar{B}$, consequently $\Pr\left[A \,\middle|\, \bar{B}\right] < \eta - \alpha$. Using conditional probabilities and the assumption that $\Pr[B] < \alpha$ we get the following upperbound on $\Pr[A]$:

$$\Pr[A] = \Pr[A \mid B] \cdot \Pr[B] + \Pr\left[A \,\middle|\, \bar{B}\right] \cdot \Pr\left[\bar{B}\right] < 1 \cdot \alpha + (\eta - \alpha) \cdot 1 = \eta,$$

but $\Pr[A] \geq \eta$, yielding the desired contradiction.

(ii) The second part of the lemma follows from Bayes theorem and the estimates we made earlier:

$$\begin{aligned}\Pr[B \mid A] &= 1 - \Pr\left[\bar{B} \mid A\right] \\ &= 1 - \frac{\Pr\left[A \mid \bar{B}\right] \cdot \Pr\left[\bar{B}\right]}{\Pr[A]} \geq 1 - \frac{(\eta - \alpha)}{\eta} = \frac{\alpha}{\eta}\end{aligned}$$ □

We can now give a proof for the actual forking lemma.

Theorem 3.33 (The forking lemma [68])**.** *Let (Gen, Sign, Ver) be a generic digital signature scheme with security parameter k. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine that within time t can produce a valid signature (σ_1, h, σ_2) over a message m of its choice with probability at least ϵ and while making at most $n > 0$ hash queries. Then we can construct an algorithm $\mathcal{B}$ that with probability at least $(\epsilon - 1/2^k)(\epsilon - 1/2^{k-1})/(16n)$ performs a succesful replay attack on adversary $\mathcal{A}$ to obtain two valid signatures (σ_1, h, σ_2) and $(\sigma_1, h', \sigma_2')$ over m with $h \neq h'$.*

Proof. We first formalize the description of the queries to the random oracle and how the choice of this oracle influences an attacher. The adversary $\mathcal{A}$ is a probabilistic polynomial time Turing machine with random input ω. It interacts with random oracle H and makes at most n hash-queries. We may assume that these queries are distinct, since, if they are not, we can make an adversary $\mathcal{A}'$ such that they are. Let $q_1, \ldots, q_n$ be the n distinct queries and $f = (f_1, \ldots, f_n)$ the vector of responses. Now, randomly choosing a random oracle (i.e. by fixing all its outputs at random) corresponds to choosing the response vector f at random. Thus for a random choice of (ω, f) algorithm $\mathcal{A}$ produces a forgery (σ_1, h, σ_2) over a message m, if it is successful.

Since (σ_1, h, σ_2) is a valid signature we know that $h = H(\sigma_1 \parallel m)$. Since f is random, the adversary $\mathcal{A}$ could have guessed this value with probability at most $1/2^k$. So it is very likely that $\mathcal{A}$ queried H for this value at some point. Let $Ind(\omega, f)$ be the index of this query, i.e. $q_{Ind(\omega,f)} = (\sigma_1 \parallel m)$ (we let $Ind(\omega, f) = \infty$ if the oracle is never queried for $(\sigma_1 \parallel m)$). We can now formally define the subset of the space of all random bit-flips and random oracles that result in a successful executions of the adversary $\mathcal{A}$:

$$S = \left\{(\omega, f) \mid \mathcal{A}^H(\omega) \text{ succeeds} \quad \wedge \quad Ind(\omega, f) \neq \infty\right\}.$$

Similarly, we define the sets S_i to be the succesful runs of $\mathcal{A}$ that make the critical hash query at position i:

$$S_i = \left\{(\omega, f) \mid \mathcal{A}^H(\omega) \text{ succeeds} \quad \wedge \quad Ind(\omega, f) = i\right\}.$$

Note that the S_i partition S. Let $\nu \geq \epsilon - 1/2^k$ be the probability that $\mathcal{A}$ is succesful and $Ind(\omega, f) < \infty$. Define the set I to be the set of most likely indices, more specifically let I consist of those indices i such that $\Pr[S_i \mid S] \geq 1/(2n)$. Then the probability that the index of succesful query lies in I is at least $1/2$:

$$\Pr[Ind(\omega, f) \in I \mid S] \geq \frac{1}{2}.$$

Let us show why. By definition of the S_i we have $\Pr[Ind(\omega, f) \in I \mid S] = \sum_{i \in I} \Pr[S_i \mid S] = 1 - \sum_{i \notin I} \Pr[S_i \mid S]$. For each of the latter terms we know that $\Pr[S_i \mid S] < 1/(2n)$ and that the complement of I contains less than n items since I cannot be empty, so $\Pr[Ind(\omega, f) \in I \mid S] \geq 1 - n \cdot 1/(2n) = 1/2$.

Let $f|_a^b = (f_a, \ldots, f_b)$ be the restriction of the responses of the oracle, f, to the elements with indices $a, \ldots, b$. We now apply Lemma 3.32 for each $i \in I$ with $X = (\omega, f|_1^{i-1})$, $Y = (f|_i^n)$, and $A = S_i$. Furthermore we take $\eta = \nu/(2n)$ and $\alpha = \nu/(4n)$. By construction $\Pr[S_i \mid S] \geq 1/(2n)$, so $\Pr[S_i] \geq \nu/(2n) = \eta$. Now let the heavy rows Ω_i be those rows such that

$$\Omega_i = \left\{ (x, y) \in X \times Y \mid \Pr_{y' \in Y}[(x, y') \in S_i] \geq \frac{\nu}{4n} = \epsilon - \alpha \right\}.$$

By Lemma 3.32 we get that $\Pr[\Omega_i \mid S_i] \geq \alpha/\epsilon = 1/2$. Let us now finally take a look at the probability $\Pr[(\exists i \in I)(\omega, f) \in \Omega_i \cap S_i \mid S]$ that a succesful forgery derives from some heavy row Ω_j for some $i \in I$. We have that

$$\begin{aligned}
&\Pr[(\exists i \in I)(\omega, f) \in \Omega_i \cup S_i \mid S] \\
=&\Pr\left[\bigcup_{i \in I} \Omega_i \cap S_i \;\middle|\; S \right] = \sum_{i \in I} \Pr[\Omega_i \cap S_i \mid S] \\
=&\sum_{i \in I} \Pr[\Omega_i \mid S_i] \Pr[S_i \mid S] \geq \frac{1}{2} \left(\sum_{i \in I} \Pr[S_i \mid S] \right) \geq \frac{1}{4},
\end{aligned}$$

so with probability $\nu/4$ the forger $\mathcal{A}$ succeeds and produces a forgery (σ_1, h, σ_2) on a message m that derives from an element $(x, y) \in \Omega_i$, i.e. that element lies in a heavy row, for some $i \in I$. Now consider what happens if we rewind the forger to the ith query and proceed with a different oracle vector f', so we have $f|_1^{i-1} = f'|_1^{i-1}$, since f is in a heavy row we know that

$$\Pr_{f'}\left[(w, f') \in S_i \;\middle|\; f|_1^{i-1} = f'|_1^{i-1} \right] \geq \frac{\nu}{4n}.$$

With probability at most $1/2^k$ we have $f_i = f'_i$, so with probability $\nu(\nu - 1/2^k)/(16n)$ we get different two forgeries (σ_1, h, σ_2) and $(\sigma_1, h', \sigma'_2)$ such that $h = f(\sigma_1 \parallel m) \neq h' = f'(\sigma_1 \parallel m)$. This gives the desired result. □

Chapter 4

Credentials systems and revocation

According to a dictionary a 'credential' is 'evidence or a testimonial concerning one's right to credit, confidence, or authority'. A slightly more general notion of credentials in the sense we will be studying was given by Chaum in his seminal paper on anonymous credentials from 1985 [26]:

> Credentials are statements based on an individual's relationship with organizations that are, in general, provided to other organizations.

Chaum introduces the concept of organizations to make the entity that supports or attests to the evidence or testimonial explicit. This matches the slightly more explicit notion of credential systems given by Camenisch and Lysyanskaya [21]:

> A credential system is a system in which users can obtain credentials from organizations and demonstrate possession of these credentials.

So to study credential systems, we also need to take into account the organizations as well as parties that solely verify credentials. In this chapter we will describe an ideal model of a credential system that allows us to describe properties of credential systems without being encumbered by practical implementation details.

Credential systems are part of a bigger, more general structure called identity management systems. This field deals with "the process and all underlying technologies for the creation, management and usage of digital identities" [1]. A digital identity corresponds to a collection of statements about a user. Verifiers want to know whether these statements are correct. In identity-management systems there are two types: network-based systems and claim-based systems.

In the former the verifier will contact the organization via the user, and verify the claims directly. We will concern ourselves here with the second, claim-based, type. Here the verifier first specifies which statements (i.e. credentials) he would like to see, the user then requests (or has previously requested) proofs of these statements from the organization and shows them to the organization. We focus on the latter because in the network-based systems revocation is trivial.

Before we continue we should point out that in real systems credentials can take very many different forms so making a general ideal model is actually useful. Traditionally a credential was synonymous with a certificate, i.e. a digital signature by an organization over a public key. Such a certificate could then have an additional field describing the statement about the holder of the public key. However, an organization may also choose to periodically publish a list of all authorized users. In order to show a credential a user only needs to prove his identify. The verifier can then check the claim by looking at the public list.

4.1 Ideal model

At the core of a credential system are the credentials themselves. However, we need to be a bit careful. A person that passed his/her drivers exam is allowed to have a driver's license, but this does not mean that this person is in possession of a driver's license. Therefore, we distinguish the subset of the users that is allowed to have a credential, i.e. passed their drivers exam in the example, from the credential objects, i.e. the driver's license.

In our ideal model we have four types of parties: users, issuers, verifiers and revocation agents. The users obtain their credential objects by interacting with an issuer. Users can subsequently show possession of such objects by interacting with the verifier. The latter checks if the user is allowed to have a credential, for example by verifying the credential object. It keeps a transcript of the conversation and asks to revocation agent to confirm that the credential object was not revoked based on this transcript. Note that we now use the term issuer instead of organization. We do this because this name is also applicable to systems that do not necessarily deal with credentials.

The transcript allows us to define various types of anonymity. Dealing with revocation is slightly more complicated, since the set of users that is allowed to have a credential then changes over time. To formalize this we introduce epoch. Each time a user is revoked (or unrevoked) we go to the next epoch. We are now ready to give a formal definition of our ideal model. A graphical representation of this system is shown in Figure 4.1.

Definition 4.1 (Ideal model of a credential system)**.** Let $\mathcal{U}$ be the set of all possible users in the system. Let C be a credential, then $S_t^C \subset \mathcal{U}$ is the set of users that is allowed to have such a credential at epoch t. Furthermore, let $R_t^C \subset \mathcal{U}$ be the set of users that at time t is no longer in S_t^C.

For each property C there exists an issuer I^C and a revocation agent A^C. All parties have access to the epoch counter t, that is initially set to zero.

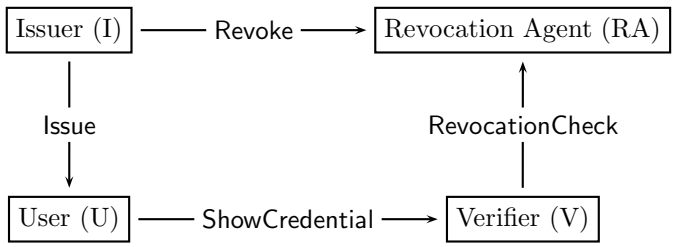


Figure 4.1: Scheme showing the four protocols between the four parties in the ideal model. The direction of the arrow indicates which party initiates the protocol. Note that we only show a revocation request by the issuer, while the model also allows these to come from the user as well as the verifier.

Issue(u, C)**:** This is an interactive protocol between a user $u \in \mathcal{U}$ and the issuer I^C for credentials satisfying property C. First, the parties set up an encrypted connection. Then user u provides all the necessary information to issuer I^C. The latter then verifies that $u \in S_t^C$. If this is the case then it cooperates with the user to generates a credential object O_u^C and sends the results to the user.

ShowCredential(O, C)**:** This is an interactive protocol between a user $u \in \mathcal{U}$ and a verifier V. The user u has a credential object O for a credential C. It converses with the verifier to convince the latter that he has such a credential object. The verifier accepts this if and only if the user u it interacts with was once granted the credential object O, i.e. $u \in S_{t'}^C$ for some $t' \leq t$ and the credential object belongs to u. Subsequently, the verifier runs RevocationCheck(T) on the transcript T of the protocol with the revocation agent A^C. If the revocation agent answers `valid` then the verifier accepts that the user has a valid credential object for credential C. Finally, it stores the transcript T of the interaction.

RevocationCheck(T, C)**:** This is an interactive protocol between a verifier V and a revocation agent A^C. The verifier asks the revocation agent to confirm that the user it conversed with in the transcript T does indeed have a valid credential object. Let u be this user. Then the revocation agent returns `valid` if $u \in R_t^C$, and `invalid` otherwise. Note that neither the revocation agent nor the verifier necessarily know u.

Revoke(u, C)**:** This is an interactive protocol between any of the parties and the revocation agent A^C. The goal of the other party is to revoke the credential object of user u for credential C. The other party provides a reason for revocation together with any necessary information (for example a proof of identity). If the revocation agent accepts the reason it adds user u to the revocation list, i.e. it sets $R_{t+1}^C = R_t^C \cup \{u\}$. Furthermore, it declares that the previous epoch has expired. Note that since the reason for revocation was valid the set of users that are allowed to have a credential object for credential C, S_{t+1}^C, is automatically updated.

4.2 Anonymity

Intuitively, to be anonymous means that you are not distinguishable from any other person. However, in practice things are a bit more subtle. For example, if you know that an anonymous post was written by a mathematics student in Groningen who wrote a master's thesis in cryptography your choices are already really limited. Nevertheless, if I reveal these both facts about myself then I can only hope for anonymity within the set of people that satisfy both facts. Hence we get the following formal definition:

Definition 4.2 (Anonymity [66])**.** Anonymity is the state of being not identifiable within a set of subjects, the *anonymity set*. Anonymity of a subject from an attackers perspective means that the attacker cannot sufficiently identify the subject within the anonymity set.

We will now study three types of anonymity as seen from our ideal model in the previous section: no anonymity, (ordinary) anonymity, and unlinkability.

4.2.1 No anonymity

Although having no anonymity is not desirable from a privacy point of view, this approach is still used the most. Consider for example the certificate, i.e. a signature over a public key, from the introduction of this chapter. Upon showing such a certificate immediately the public key, and hence the identity of the user, is revealed. In our ideal model we would then have the following

Definition 4.3 (Non-anonymous credential system)**.** A credential system is non-anonymous when any entity can recover u in probabilistic polynomial time from a transcript T of an ShowCredential interaction between user u and some verifier. In other words, we can assume that the transcript T contained u directly.

We will see more extensive examples of traditional systems that are non-anonymous in Section 5.1.

4.2.2 Anonymity

A first real notion of anonymity is the following: given a transaction it should be impossible to recover the identity of the user. This is not really difficult to achieve. If a verifier can only recover a pseudonym of a user given a transaction then already this user is anonymous in the sense of Definition 4.2, as long as no one has a map that translates pseudonyms to identities it is not possible to recover the identity of a user.

For credential systems we then arrive at the following definition.

Definition 4.4 (Anonymous credential system)**.** A credential system is anonymous when no entity can recover u in probabilistic polynomial time from a transcript T of an ShowCredential interaction between user u and some verifier.

The anonymous OV-Chipkaart in the Netherlands is a good example. While the card has a unique identifying number this number is not linked to the

identity of a person. Another example is the pseudonymous system proposed by Chaum [26]. In this system users use new pseudonyms with every issuer and every verifier. The issuers and verifiers only ever learn the pseudonym the user originally used with them. Hence the identity of the user remains hidden. Furthermore, the pseudonyms by which the user is known to other issuers/verifiers are hidden as well. Note that using multiple pseudonyms or identifying pieces of information gives a slightly stronger notion of anonymity than using only one for every party.

We note that for real world applications this level of anonymity might not be strong enough. Given a complete lists of a person's financial transactions it is, in general, rather easy to figure out the true identity of this person.

Interestingly, a related concept shows up in group-signatures where a designated authority is able to examine a signature and trace it to the person that produced it. In other words, given a transcript, it is able to recover the identity of the user. Since this is in fact the inverse of what we described above, we feel that untraceability should equal anonymity, and not unlinkability, as some authors propose [77].

4.2.3 Unlinkability

Unlinkability is the holy grail of anonymity. When two credential showings are unlinkable an adversary has no method for deciding whether they originate from the same user or not. Thus every transaction seems to be completely unrelated. If a system is traceable, then surely it is also linkable, hence unlinkability is a stronger assumption than the (ordinary) anonymity we defined earlier. Formally unlinkability is defined as follows.

Definition 4.5 (Unlinkability [66])**.** Unlinkability of two or more items of interest from an attackers perspective means that within the system, the attacker cannot sufficiently distinguish whether these items of interest are related or not.

An unlinkable credential system is then described as follows.

Definition 4.6 (Unlinkable credential system)**.** A credential system is unlinkable when no entity can determine in polynomial time whether for two transcripts T_1, T_2 between users u_1, u_2 and verifiers V_1 and V_2 the users u_1 and u_2 are equal or not.

In the next chapter we will see many examples of schemes satisfying unlinkability. In addition to this, we will prove that our scheme is unlinkable in Chapter 7.

4.3 Revocation

Revocation is the act of recall or annulment and is an essential part in credential systems. In practical situations a credential may need to be revoked because it is stolen, in which case the user will revoke his credential to prevent abuse by other parties. Similarly, an issuer may decide to revoke a credential because of

fraud or because the conditions upon which the credential was issued are no longer valid.

In our ideal model a user can run ShowCredential while his credential has been revoked. The only consequence is that the verifier detects the cheating. In many practical systems also the identity of the user is revealed. Depending on the type of system this is desirable or not. We feel that in general this is not a problem since a revoked credential should not be used. However, the following definitions on revocability are independent of whether the identity is revealed or not.

In the real world model the verifiers are provided with some additional information that allows them to test whether a credential is revoked. Suppose this additional information is always the same. Then verifiers have a method that, given a new transaction and this additional information, decides whether the credential is revoked. This method often also works on transcripts of interactions that occurred before revocation took place, hence revealing the entire history.

Definition 4.7 (Weakly private revocability)**.** We call revocability weakly private if revoking a the credential C of a user u in epoch t does not only invalidate any future uses of the credential object in epochs after epoch t, but will also make all transactions of user u for credential C linkable. More precisely, given a transcript T of the interaction of user u showing credential C to a verifier there is a polynomial time algorithm that determines if a transaction T' is for the same user and same credential.

Consequently, we also have a stronger version of revocability that is, for obvious reasons, often called backwards unlinkable.

Definition 4.8 (Backward-unlinkable revocation)**.** We call revocability in a credential system *backward-unlinkable* if revoking a the credential C of a user u in epoch t does not only invalidate any future uses of the credential credential object in epochs after epoch t, but will also make all transactions of user u for credential C after epoch t linkable. More precisely, given a transcript T of the interaction of user u showing credential C to a verifier there is a polynomial time algorithm that determines if a transaction T' is for the same user and same credential if and only if both transactions occur after epoch t.

Finally, we can define revocation that offers the most anonymity for revoked users. Here, no linking is possible, even though a credential for a user has been revoked.

Definition 4.9 (Anonymous revocation)**.** We call revocability anonymous if revoking a the credential C of a user u in epoch t invalidate any future uses of the credential credential object in epochs after epoch t, and in addition no linking of transaction of this user for this credential are possible, even though the credential has been revoked.

Revocation in the model we described is at the credential level. In some situations it might be beneficial to revoke a complete user instead. Some systems support this type of revocation.

We did not specify how the revocation agent can detect the identity of the user it wants to revoke. It could possibly obtain this information by contacting a trusted store, but whether this will work in practice is highly dependent on the implementation of the credential system. Therefore we will discuss the method of revocation while describing the protocols themselves.

4.4 Unforgeability

A natural security requirement for a credential system is to ask that no user can create his own credentials. For a real world model we should be a bit careful in how we define unforgeability. In the simple case, where an adversary is not allowed to interact with valid users, it is easy to define unforgeability: the adversary should not be able to demonstrate possession of a credential.

The case where an adversary controls multiple users is more difficult to analyze. Even in the ideal model. Suppose in the real world that an adversary extracts the knowledge of a number of users. The adversary can now always use one of their credentials. This is unavoidable. The adversary should, however, not be able to construct a new credential that is really different from the credentials it has extracted. In the ideal model this is captured by the verification by the verifier. This only succeeds if a user shows his own credential object (or perfectly impersonates another user and shows that users' object). This is not the case for a new credential object.

In a number of protocols, most notable the group-signatures and our own protocol, we can use the revocation mechanism to define what it means that the adversary cannot create new credentials, even after interacting with or subverting a number of users. Recall that after revocation the use of a revoked credential could be detected. So, an adversary can only build a really new credential if it can show a credential while all the users it subverted are revoked.

4.5 Real-world model

In the real world the verification done by the verifier should be made easily implementable. In addition revocation check does not always need to use an online revocation agent. The model we sketch here reflects the structure of the systems in the next chapter. There also exists other systems that follow a different structure (for example the user always authenticates via an issuer) but we will not consider these here.

In real world systems we again find users (U), issuers (I) and verifiers (V) with the same role as in the ideal model. Some protocols use an online validity check on credentials. That is why we split the revocation agent into two parts: one offline (RA) and one online revocation status agent (RS). The offline part handles the actual revocation of credentials and distributes this information to the other parties. The online agent can be used a a third party that certifies validity of a certificate during the showing processes.

All the relevant parties and how they interact are shown in Figure 4.2. Not all these interactions are necessary in any system. For example the online revocation status agent is not used in many systems, but it is used in the Online

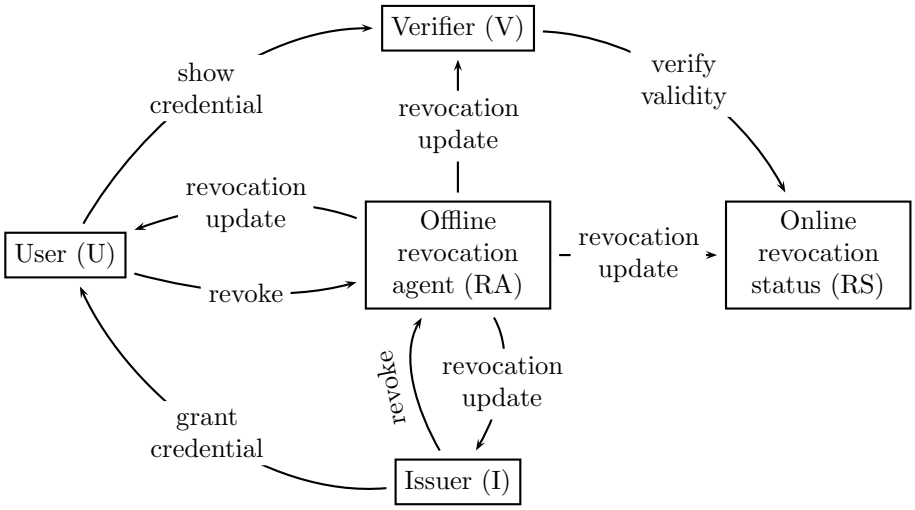


Figure 4.2: The important parties in a credential system with the protocols that can be run between them. The origin of the arrow denotes the initiating party. Note that not all these protocol (nor parties) have to be supported by a credential system.

Certificate Status Protocol, see Section 5.1.2. The revocation agent responds to a revocation by sending the necessary updates to the other parties. In most protocols we shall consider in the next chapter only the verifier is notified. However, for example when dealing with accumulators, see Section 5.4, also the users' certificates need to be updated. In practice multiple issuers and multiple verifiers may be present in the system.

4.6 Aspects of credential systems

To compare various credential systems we need to define aspects on which to judge them. With respect to privacy we establish whether the system is totally non-anonymous, untraceable or unlinkable. We will consider computation aspects, i.e. how much work need the various entities perform, and communication aspects, i.e. how much data needs to be transferred between which entities. We consider each of these in turn.

4.6.1 Computation aspects

As we saw in the introduction of this thesis smart cards are very useful in protecting valuable (key) information. This makes them very interesting as a personal wallet that contains your credentials. However, they are only equipped with an 8-bit processor running at a slow clock speed. Hence, they are rather slow. Additionally, both the semi-permanent storage for key material (stored in an EEPROM) and the working memory are small as well. Combining this with the fact that showing a credential should be very fast shows that the amount of computation that is performed and the storage used by the user should be very

limited. This will be the first computational aspect we will consider. Ideally we want a constant time showing-algorithm for the users.

The time limit on transactions also impacts the verifiers and issuers. When the revocation check is very costly this will also cause long verification times. So the effect of the length of the revocation list is another important point of interest. Ideally we want a constant or logarithmic time dependency on the length of the revocation list, but we shall also see protocols where this dependency is linear.

4.6.2 Communication aspects

Just as the computation speed of a smart cards is limited, so is its communication speed. Contactless smart cards can achieve data rates from 108 to 848 kbit/s. This means that the amount of data these cards can send in the short time allowed for a transaction is also limited. The communication between the user and the verifier (issuer) is the first communication aspect we will consider.

On the other hand there is the issue of scale. While revocation is not a very frequent event, it does impact the system if each time a credential is revoked all users need to contact the (offline) revocation agent. This might result in a performance hit if the number of users grows. That is why the communication between the users and the revocation agent is our second communication aspect. Ideally the users do not need to be notified of revocations.

Finally, we consider the amount of communication between the revocation agent and the verifiers/issuers. Here we are more interested in the order of the communication. Is the amount constant for each update, or is there maybe a linear dependency on the current number of revoked credentials. The (optional) communication between the verifier (issuer) and the online revocation agent is constant for all protocols we examine and therefore not really interesting beyond whether it is used or not.

Chapter 5

Revocation Techniques

In this chapter we review a number of credential systems, while focussing on their revocation abilities. In addition we examine a couple of concepts that may be added to existing credential systems and hence provide revocation. We first examine a very traditional credential system: the X.509 certificate system that is used throughout the internet. This system demonstrates some nice concepts such as limited life-times and revocation lists. We continue with more recent credential systems in Section 5.2. In Section 5.3 we study group signatures, which can be converted into a credential system, while in Section 5.4 we study accumulators which can be used for revocation checking. Finally we study private set intersections, that also have some possible applications to revocation checking in Section 5.5.

5.1 Traditional methods

The X.509 ITU-T [49] standard describes a complete public key infrastructure. As we have seen before, in public key systems every entity has a public and a private key. To verify that an entity signed a message we look up its public key and check the signature against this public key. This lookup is a weak point of the system: our security guarantees only hold if this lookup is infallible. Hence we have to be sure that the public keys indeed corresponds to the correct entity. In the X.509 system this is solved in a hierarchical fasion.

The system contains organizations, here called Certificate Authorities (CAs), that can sign statements about entities, or, more precisely, public keys. These statements are called certificates. Verifiers can then check these certificates against the public key of the certificate authority. By signing a certificate the CA links the identity of an entity to a public key. The identity of this certificate authority is subsequently certified by a higher level authority, thus leading to a hierarchical structure.

In Figure 5.1 you can see a X.509 certificate. Since this is a practical and rather general system this certificate constrains more information than the

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Bacca,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=bacca@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                   70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                   16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                   c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                   8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
       d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
       0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
       5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
       8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
       68:9f
```

Figure 5.1: Example X.509 certificate by the CA Thawte Consulting for the public key of www.freesoft.org. Notice how the signature authenticates the complete Certificate record, including expiration dates.

bare minimum. The most important fields are the Subject Public Key Info field containing information about the public key and the Signature Algorithm field containing a description of the signature algorithm and the resulting signature. This signature is over the whole preceding text. From a semantic point of view also the Subject field is interesting. The signature of the certificate authority binds the identity of the person described in this field to the public key.

When using a certificate like this both parties have to be careful. First, the holder of this certificate cannot hope to remain anonymous. Even if the Subject field would only contain a pseudonym any usage of this certificate will remain traceable. Second, also the verifier of such a certificate needs to be careful. Being able to show this certificate does not necessarily also mean owning the certificate. The verifier needs to convince himself that the shower also knows the corresponding private key. It could do so by asking the holder of the certificate to sign a random nonce, although this has its issues as well.

The Validity field limits the lifetime of a certificate. If a system does not support any other forms of revocation then this can be used as a substitute especially when a short expiration time is specified when the certificate is issued. If a certificate is revoked the holder simply does not get a new one. While easy to implement this approach also has a couple of downsides. The first is the window of vulnerability. After revocation a certificate can still be used until it expires. To counteract this we have to choose short – the exact definition of short depends on the context – expiration time. This brings us to the second problem. If the life-time of a certificate is small then they need to be reissued very often. This results in a computational burden on both the organizations (CAs) as well as the users. In fact, if the average time interval between showing a certificate is larger than the life-time of a certificate then a user needs to obtain a new certificate each time it wants to show it. Hence the system has been transformed into an online system.

5.1.1 Certificate revocation lists

The X.509 standard does specify a better system for revocation of certificates: certificate revocation lists (CRL) [48]. These lists are timestamped, periodically issued and signed by the certificate authority. They are stored in a public repository. A certificate is added to the CRL by including the serial number. When a certificate is shown its serial number is visible, so a verifier can check in $O(\log n)$ time, where n is the length of the CRL, if the certificate has been revoked.

Once a certificate has been added to the CRL it has to remain there until the certificate itself expires. This ensures that the CRLs do not become too large. So, even though the expiration time is not used to enforce revocation it does affect it.

A CRL is a classical example of a *blacklist*. The CRL contains those certificates that are no longer valid. We shall also see other types of systems that turn this approach around: a certificate is only valid when it is on the list. Such a list is called a *whitelist*. In Section 5.4 we shall see an especially nice example where these whitelists can be kept very small.

5.1.2 Online Certificate Status Protocol

The CRLs we just described are not without their problems. For example the revocation list might be too big for the verifier to retrieve. That is why the Online Certificate Status Protocol (OCSP) [61] was developed as an online alternative. Instead of locally checking revocation status against the CRL a verifier will now send the serial number of the certificate to the OCSP server. The latter will verify the validity of the certificate. If the certificate is still valid it sends a signed success message back.

One of the clear advantages is that a verifier now does not need to store the complete list, but can query for the status of a certificate online. This does mean a larger load for the OCSP servers, but they can easily be distributed to reduce this. The name of the OCSP server can be added to the certificate so the verifier doesn't need to look this up additionally. The primary disadvantage of online protocols is that they need a working link between the verifier and

the OCSP server. For our credentials this is not an acceptable situation since this would mean that certificates cannot be checked when this connection is abscent, which is why we will not consider online protocols further.

5.2 Existing anonymous credential systems

In this section we study the powerful anonymous credential systems Idemix, short for identity mixer, and U-Prove. Their approaches are a bit different: credentials from Idemix can be used as often as one wants, while U-Prove credentials are in principle single-show. However, the construction of U-Prove is simpler and more efficient.

5.2.1 Idemix

The Idemix credential system, proposed by Camenisch and Lysyanskaya [21] is more formal than our ideal model from the previous chapter and instead follows a model formerly proposed by David Chaum. In order to obtain a credential object from an issuer a user first obtains a pseudonym from this issuer and will subsequently use this pseudonym in any further communication with the issuer. Also, credentials are issued with respect to these pseudonyms.

For simplicity we will only consider a single issuer. For this issuer we fix an RSA-modulus n and five elements a, b, d, g, h in $QR(n)$, the group of quadratic residues modulo n. Every user U has a master key x_U, that is never revealed. The implementations of the transactions are heavily based on zero-knowledge proofs and quite complicated. We give a simplified exposition of these protocols, for the details we point to the original paper [21].

The user and the issuer first jointly agree on a pseudonym $N_U = N_1 \parallel N_2$, the concatenation of a user-generated part N_1 and an issuer-generated part N_2 using an interactive protocol. This pseudonym is accompanied by a validating tag P_U. This tag is a Pedersen commitment, see Definition 3.16, to the master key x_U of the user: $P_U = g^{x_U} h^{s_U}$, where s_U is a random value that is jointly computed by the user and the issuer and only known to the user. This ensures that the value of P_U is statistically independent from x_U. The complete pseudonym-issuing protocol is quite complicated because the user needs to prove that it constructed the validating tag correctly, i.e. by using the randomness provided by the issuer. We refer to [21, Protocol 1] for the details.

A credential on a pseudonym N_U is a tuple (e_U, c_U) such that $c_U^{e_U} = P_U d \pmod n$. By the assumption that the strong RSA problem is hard such tuples cannot be forged.

Problem 5.1 (Strong RSA Problem (sRSA) [75])**.** Given an RSA modulus n, and a random element $z \in \mathbb{Z}_n^*$ find $e > 1$ and u such that $z = u^e \pmod n$.

The Issue protocol is implemented as follows. First user U sends (N_U, P_U) to the issuer and authenticates herself by executing the following proof of knowledge:

$$PK\{(x_U, s_U) : P_U^2 = (a^2)^{x_U} (b^2)^{s_U}\},$$

where the additional squaring of both sides proves that the elements are quadratic residues. The issuer ensures that the pair (N_U, P_U) matches its own

database. Then it chooses a random prime e_U and computes $c_U = (P_U d)^{1/e_U} \pmod{n}$. It can do this because the issuer knows the factorization of n. Finally it sends the credential (c_U, e_U) to the user. Both parties store it for future reference.

Finally we have the showing protocol, ShowCredential. A user U has a credential when she can show that she knows values (P_U, c_U, e_U) satisfying the equations as before. However, revealing any of them would void anonymity, so we cannot use the normal approach where we reveal one of them and then show in zero-knowledge that we know the rest. Therefore, user U and verifier V engage in the following protocol. First, U chooses r_1, r_2 at random, computes Pedersen commits to c_U and r_1: $A = c_U h^{r_1}$ and $B = h^{r_1} g^{r_2}$, and sends the commitments to V. Since r_1, r_2 are random this does not reveal anything about the actual value of c_U but does allow us to write down a proof of knowledge between U and V that shows that U also knows e_U and P_U:

$$PK\{(e_U, x_U, s_U, r_1, r_2, \alpha, \beta) : d^2 = (A^2)^{e_U} (\tfrac{1}{a^2})^{x_U} (\tfrac{1}{b^2})^{s_U} (\tfrac{1}{h^2})^{\alpha} \wedge B^2 = (h^2)^{r_1} (g^2)^{r_2} \wedge 1 = (B^2)^{e_U} (\tfrac{1}{h^2})^{\alpha} (\tfrac{1}{g^2})^{\beta} \wedge x_U, s_U, e_U \text{ have correct ranges}\}.$$

The variables α and β represent the values $r_1 e_U$ and $r_2 e_U$ respectively. The range checks are necessary because the values of x_U, s_U, e_U are restricted in magnitude. There exists Σ-protocols for proving these things. An even more complicated zero-knowledge proof can be used to show possession of some pseudonym with another issuer.

Idemix can additionally provide protection against fraud using two different methods. A major problem with digital credentials is that they can be easily duplicated, thus allowing for easy transfer of credentials. As a deterrence against this Camenisch and Lysyanskaya propose *all-or-nothing disclosure*: sharing some information, say a credential, in fact reveals all information about you. This idea is already present in the protocol we described above. For every showing of a credential the secret key x_U is needed. To get all-or-nothing disclosure the system can be extended such that some very valuable information, like access to a bank account, is encrypted using x_U and stored in a public place. Then, when sharing a credential, also this valuable information is accessible.

Idemix can also be extended to allow for anonymity-revocation. During the show protocol the verifier is additionally provided with a verifiable encryption of the private key or pseudonym of the user. A trusted third party can then reveal the identity of the user if a suitable condition is met. This can be used to implement revocation in the sense of the previous chapter but it is not very efficient, since the trusted third party needs to decrypt a message for every verification, nor is it very privacy friendly, as the trusted third party can then trace any interaction.

5.2.2 U-Prove

Idemix uses complicated credentials, that are shown using equally complicated zero-knowledge proofs. The U-Prove credentials [18] on the other hand are much simpler. In addition, they can easily be extended to support multiple

attributes, for example a name, date of birth, place of birth and sex. The owner of such a credential can opt to reveal only part of this information. Finally, U-Prove credentials can in principle only be used once.

Let G be a multiplicative group of prime order q chosen by the organization. This group can for example be an elliptic curve, or a subgroup of $\mathbb{Z}/n\mathbb{Z}$. We follow the original publication in writing this group multiplicatively. Suppose a user has a number of attributes $(x_1, \ldots, x_l)$, all modulo q, that the organization wants to encode in a credential. The organization publishes a generator h_0 of G as well as a generator g_i for every attribute.

We examine a simplified version of the protocol. A credential on attributes $(x_1, \ldots, x_l)$ is given by the public key

$$h = g_1^{x_1} \cdots g_l^{x_l} h_0^{\alpha},$$

with corresponding private key $(x_1, \ldots, x_l, \alpha)$ and a signature σ over h by the organization. Note that h is a generalization of the Pedersen commits we saw in Section 3.3.2. The randomness of α ensures that h does not leak any information about the attributes encoded within, in fact, for every choice of $(x'_1, \ldots, x'_l)$ there exists an α' such that

$$h = g_1^{x'_1} \cdots g_l^{x'_l} h_0^{\alpha'}.$$

This also means that to cheat, i.e. to use σ as a signature for other attribute values, a user has to find this exact value of α', which corresponds to solving a discrete logarithm problem.

The most simple usage of this credential would be to use it just as a classical certificate. The user provides the verifier with h and σ and shows that it knows the private key using a simple proof of knowledge of discrete logarithms. To guarantee anonymity a credential of this form can only be used once. But, that could also be achieved using a simpler scheme. They shine, however, when a user selectively discloses a number of attributes. Consider, as an example a credential $h = g_1^{x_1} g_2^{x_2} h_0^{\alpha}$. Suppose user U wants to disclose x_1 but not x_2. First, he reveals h, x_1 and σ. Then he constructs a derived credential $h' = h/g_1^{x_1} = g_2^{x_2} h_0^{\alpha}$ and proves in zero-knowledge that he knows the remaining private keys

$$PK\{(x_2, \alpha) : h' = g_2^{x_2} h_0^{\alpha}\}.$$

The complete protocol is shown in Figure 5.2. Note that this protocol is just a more generalized version of the Prove by Schnorr we saw in Section 3.12. Just as with other Σ-protocols the interactive proof can be converted into a signature scheme using the Fiat-Shamir heuristic.

The U-Prove credentials can be used in even more flexible ways. For example it is possible to prove linear relations over the attributes, as well as logical expressions involving binary attributes. These properties make U-Prove credentials very attractive as general purpose solution.

In the base scheme a user can show a credential h more often, but these instances are then traceable. However, it is also possible to make one-show certificates. These are useful to construct for example virtual money. Recall that Σ-protocols satisfy the special soundness property. Thus if a user uses the

User		*Organization*
$r_2, r_3 \in_R \mathbb{F}_p$		
$R = g_2^{r_2} h_0^{r_3}$		
send h, σ, R, x_1	$\longrightarrow$	**into** $\bar{h}, \bar{\sigma}, \bar{R}, \bar{x_1}$
		challenge $c' \in_R \mathbb{F}_p$
into c	$\longleftarrow$	**send** c'
$s_2 = cx_2 + r_2$		
$s_3 = c\alpha + r_3$		
send s_2, s_3	$\longrightarrow$	**into** $\bar{s_2}, \bar{s_3}$
		verify $h^{c'} \bar{R} = g_1^{\bar{x_1}} g_2^{\bar{s_2}} h_0^{\bar{s_3}}$

Figure 5.2: Showing protocol of U-Prove for a credential ($h = g_1^{x_1} g_2^{x_2} h_0^{\alpha}, \sigma$) where the value of x_1 is revealed while the value of x_2 is kept hidden. The send-into notation used here serves to distinguish what one party sends, from what the other party receives. For example, the user sends R, but the verifier will refer to this value as $\bar{R}$.

same commitment R twice then the secrets can be recovered. By fixing a single R as part of the signature σ by the organizations the user can only use this commitment. If it is used a second time all attributes can be recovered. By including some personal information like a social security number or a credit card number as an attribute abusers are easily traceable.

Deterrents like all-or-nothing disclosure are also possible in U-Prove: simply include an attribute with this secret information in the credential. If someone else wants to use it then they need to know this secret information.

In order to preserve anonymity a credential can be used only once (since the signature is always the same). This makes it easy to implement revocation. After a user is revoked the organization will of course not issue any new credentials. Furthermore, it can add its signature σ to a revocation list that can then be checked just as easily as in the traditional protocols.

5.2.3 Efficiency and conclusions

Intuitively the zero-knowledge proofs of Idemix are a bit more involved then the proofs in U-Prove. In fact, if we consider the single bit of information that states that a user has a credential then we have a U-Prove credential with no attributes. Consequently, the protocol for U-Prove is just a Schnorr proof of knowledge of a discrete logarithm. In the following analysis we focus on the simple showing protocols and consider the computational complexity of the user.

In 2007 Danes [32] performed a thorough analysis of the various operations that are needed to complete the showing protocol of Idemix. A comparison is shown in Table 5.1. The number of exponentiations counts the number of operations in the cyclic group G of prime order p, while the number of multiplications counts the multiplications modulo p. Note that even though U-Prove (in the simplest form) is quite fast it is not really suited for smart cards since a credential can only be used once. Furthermore, in order to have

	Nr. exponentiations	Nr. multiplications	Duration
Idemix	10	8	~ 10 sec [11]
U-Prove	1	1	~ 2 sec

Table 5.1: Comparison of exponentiations in group G of primer order p and multiplications modulo p for the showing protocols of Idemix and U-Prove. The durations denote the time needed for these protocols on a smart card. The duration of U-Prove is estimated based on [32] and [5]. Very recently Mostowski and Vullers showed that the U-Prove running time could be further reduced to be less than 1 second even when dealing with multiple attributes [60].

revocation users should request the credentials right before they use them, so this incurs a high computational cost.

5.3 Group signatures

A group signature is a special type of signature that can be created by any user in a specific group, but that traces to the group as a whole. So a group signature represents a signature by the group as a whole. This notion was first introduced by Chaum and Van Heyst in 1991 [27]. Such a scheme is quite useful as a basis for anonymous credentials. Every user who can sign a message, say a random nonce, on behalf of the group is said to have the credential belonging to that group.

An important property of a group signature scheme is that the identity of the signer cannot be inferred from the signature (anonymity), except by a special entity that has a special tracing or opening key (traceability). Besides the users, various other entities play a role in group signature schemes. The group manager (GM) is responsible for providing keys and certificates on those to the users. Furthermore, the GM is responsible for revocation. In a credential system the group manager corresponds to the issuer. The tracing manager (TM) is capable of retrieving the identity of the signer. In practice, the same entity may simultaneously function as the group manager as well as the tracing manager.

It is interesting to study these group signatures with the intention to use them as a credential system. In most group signatures members can be added to this group, i.e. given a credential, without any additional cost. Also, revocation has been included in most of them. The one downside is that the traceability property might not be desirable from a privacy point of view. However, traceability might be removed from some protocols where it is not directly used for revocation.

A detailed analysis of the formal properties of group signatures is provided in [9]. For some practical applications of group signatures properties like non-framability and exculpability are important. These properties ensure that if a signature opens to a user then it is known to be produced by that user (and not forged by for example the group manager), respectively the user can prove he did not create that signature if he did not. For our purpose of studying revocation these are of less importance, so we will not consider them further.

The standard method for creating a group signature scheme is to provide each member with a signed private key that will verify against the group public key. Opening is subsequently supported by requiring that the members supply a verifiable encryption of their private key. The tracing manager can decrypt this cipher text to open a signature and obtain the identity of the signer.

The first reasonably practical application of group signatures with revocation is due to Ateniese et al. [3] which is based on the ACJT scheme [2]. It was also the first that was provably secure as well as reasonably efficient, i.e. the size of the signatures is independent on the group size. We will briefly describe this scheme. Furthermore we will describe the recent scheme by Chen and Li [29] that uses bilinear mappings to achieve better results.

5.3.1 Formal definition

Definition 5.2. A group signature scheme consisting of the following procedures:

Setup(1^k)**:** A setup procedure run by the group manager resulting in a group public key gpk and a group manager secret key gsk.

Join$()$**:** An interactive protocol run by the GM and a user U_i. After running this protocol the user learns a secret key x_i and a certificate c_i. Furthermore, the GM stores an entry t_i in the tracing database for access by the TM.

Sign(gpk, x_i, c_i, m)**:** On input of public group key gpk, a private key x_i, certificate c_i and a message m the procedure produces a signature σ on the message m.

Verifiy(gpk, m, σ, RL)**:** On input of a public group key gpk, a message m, a signature σ on that message and a revocation list RL the procedure verifiers whether the signature is valid given the group key and the revocation list.

Open(gpk, m, σ, TDB)**:** Given gpk, a message m, a signature σ on that message and the tracing database TDB, the TM returns the identity of the user that created the signature.

The exact definition of security depends on the specific implementation of the group signature. The various security properties can be defined very precisely in terms of security games, see Bellare et al. [9] for a rigorous definition. The following list gives a sketch of the most often used security definitions [9, 29]:

Correctness Any signature σ produced by Sign will be verified by Verify, unless the user appears on the revocation list.

Anonimity We define anonymity in terms of a security game. In fact this security game is very similar to the security game for indistinguishability of encryptions in Definition 2.7. Now the goal of the adversary is not to distinguish two ciphertexts but two group signatures. In the find phase the adversary is allowed to do anything it wants to the system: adding users, obtaining signatures from the users on any message, corrupting users (i.e. obtain their private keys) and opening messages. Then it produces a message and selects two users, of which is does not have the private key and that are not revoked, on which it wants to be challenged.

The challenger provides the adversary with a signature from one of these users. The adversary wins if it can correctly guess which user produced the signature in more than half of the cases.

Traceability Given a valid signature, i.e. a signature that validates according to Verify, the tracing manager will always be able to recover the identity of the user that created the signature. In the case of collaboration between users at least one of the collaborating users' identities will be revealed.

Note that this type of anonimity conforms to our definition of unlinkability in Definition 4.5.

5.3.2 ACJT scheme

This scheme by Ateniese et al. [3] uses a number of standard techniques. Its security is based on the hardness of the Strong RSA and Decisional Diffie-Hellman problems. During the setup phase the group manager generates a safe RSA modulus $n = (2p'+1)(2q'+1)$ and a number of elements $a, a_0, g, h \in QR(n)$, the group of quadratic residues modulo n. In this scheme the group manager and the tracing manager are the same entity.

Members will encrypt their siging key in such a way that the group manager can decrypt it. To this end, the group manager randomly generates a private key $x \in_R \mathbb{Z}/(p'q')\mathbb{Z}$, and calculates the corresponding public key $y = g^x$. In addition, the group manager keeps p', q' as private keys, all other values are made public.

Upon joining, a member receives a certificate (A_i, e_i) such that $A_i = (a^{x_i} a_0)^{1/e_i} \pmod n$, where x_i is the private key of the user. A signature on a message m is obtained by applying the Fiat-Shamir heuristic to a proof of knowledge of knowing the certificate. Just as with the Idemix scheme we need to blind the certificate (A_i, e_i) as revealing it voids anonymity. To do so pick $k \in \mathbb{Z}/(p'q')\mathbb{Z}$ and set

$$T_1 = A_i y^k, \qquad T_2 = g^k, \qquad \text{and} \qquad T_3 = g^{e_i} h^k.$$

Subsequently a signature on a message m is obtained by the Fiat-Shamir heuristic on the following proof of knowledge:

$$\sigma' = SPK\{(k, e_i, x_i, \alpha) : a_0 = \frac{T_1^{e_i}}{a_i^x y^\alpha} \wedge 1 = \frac{T_2^{e_i}}{g^\alpha} \wedge T_2 = g^k \wedge T_3 = g^{e_i} h^k\}(m).$$

This proof of knowledge proves two things: (1) the pair (T_1, T_2) is en ElGamal encryption of A_i under the public key y, and (2) the value e_i belongs to A_i.

Ateniese et al. suggest two interesting revocation methods. In the first the group public key is changed and every user receives an update to adapt his certificate. The updates are crafted in such a way that revoked users cannot apply the update, and hence no longer have a valid certificate. In some sense this method forms an implicit whitelist. The problem with this approach is that the number of updates are linear in the number of remaining users, and quite inefficient. Furthermore, each time a user is revoked every user has to do an update. Later Camenisch and Lysyanskaya [22] and Boneh et al. [14] created more efficient updating schemes.

The second approach moves computational load to the verifier. The group manager maintains a list of revoked users, the revocation list. Since users on the list are no longer valid, this is a black list approach. To allow the verifier to check whether they are on the list users provide a blinded piece of personal information. Let $G = \langle \tilde{g} \rangle$ be a group of order n. Suppose users $U_1, \ldots, U_m$ should be on the revocation list. The group manager first generates a random element $b \in_R QR(n)$, and for each U_i adds b^{e_i} to the revocation list. In the signing proof of knowledge users provide two additional values

$$T_4 = f = \tilde{g}^r \qquad \text{and} \qquad T_5 = f^{(b^{e_i})}, \tag{5.1}$$

and proves that they are well-formed. This requires a proof of knowledge of a double discrete logarithm for e_i. These proofs are very inefficient, requiring around 500 exponentiations for a reasonable security parameter. The verifier can easily compare every element x on the revocation list to T_5 by calculating f^x. Note that even in this scheme all users need to receive the generator b that is currently used for the revocation list.

5.3.3 Verifier Local Revocation

Recently, several group signature schemes have been created based on bilinear maps [14, 17, 23, 29, 39, 62]. Of those, the schemes proposed in [17, 29, 39] support verifier local revocation (VLR). With this type of revocation, changes made to the revocation information needs only be sent to the verifiers. The verifier can then use this information to determine if a signer has been revoked. Boneh and Shacham [17] note that given a VLR scheme tracing is easy. Verifying a signature using a singleton revocation list containing each user in turn will immediately indicate the user that generated the signature.

We will study the scheme by Chen and Li [29] in a bit more detail. During the setup phase the group manager generates a bilinear map $e : G_1 \times G_2 \to G_T$ and G_1, G_2, G_T cyclic groups of prime order p with g_2 is a generator for G_2 and $g_1 = \phi(g_2)$ a generator for G_1 with $\phi : G_2 \to G_1$ a homomorphism from G_2 to G_1. We really need this homomorphism, so we can only use type 1 and type 2 pairings, see Section 3.5 for details. The signature produced by the GM is based on the q-SDH problem first introduced by Boneh and Boyen [12], see Definition 3.28 and Section 3.6.2. Following the original notation we write the groups multiplicatively.

During the setup phase, the group manager also generates elements $h_1 \in_R G_1$, $h_2 \in_R G_2$ and $\gamma \in_R \mathbb{F}_p^*$. The value γ will be the private key of the group manager, while $w = g_2^\gamma$ is the corresponding public key. In order to speed up calculations the following values can be precomputed:

$$T_1 = e(g_1, g_2), \quad T_2 = e(h_1, g_2), \quad T_3 = e(h_2, g_2), \quad \text{and} \quad T_4 = e(h_2, w) = T_3^\gamma.$$

A user joins the group by engaging in the Join protocol with the group manager. The user selects a secret key $f \in \mathbb{Z}_p$ and computes the corresponding public identity $F = h_1^f$. Then the user convinces the group manager that F is well-formed using a simple proof of knowledge. The group manager generates $x \in_R \mathbb{Z}_p$, computes $A = (g_1 F)^{1/(x+\gamma)}$ and sends both to the user. Furthermore, it stores (F, x) in the tracing database. Note that the GM never learns f.

During the signature process the user generates $B \in_R G_1$ and computes blinded values of f and x: $J = B^f$ and $K = B^x$. Furthermore, he/she creates a blinded version of A: $T = Ah_2^a$. He/she then proves knowledge of such x, f corresponding to J and K, and, furthermore, shows that A is formed using this x and f. A signature is obtained by the Fiat-Shamir heuristic on the corresponding proof of knowledge:

$$SPK\{(a, b, x, f) : J = B^f \wedge K = B^x \wedge e(T, w) = e(T, g_2)^{-x} T_2^f T_3^b T_4^a\}(m).$$

Note that this proof guarantees that the x embedded in T and hence A is the same as the one embedded into K. Users are revoked by by adding their private key x to the revocation list. After the signature has been confirmed, the verifier checks for each $x' \in RL$ if $K \neq B^{x'}$. If a match is found, then the user was revoked, and the signature should not be accepted. Note that this ensures that any previous signature by this user is traceable as well.

5.3.4 Alternative constructions

Besides the two constructions we covered in detail, a number of alternatives has been proposed. The scheme by Furakuwa and Imai [39] is also based on bilinear maps and the hardness of the SDH problem, but it does not support verifier local revocation. However, in this protocol the group manager and the tracing manager are allowed to be different entities: while the tracing manager can open a message, the group manager can not. This seperation provides better protection against corruption of the GM and/or TM, see Bellare et al. [9].

Another alternative is proposed by Nakanishi and Funabiki [62], this scheme is an extention of [17] that solves the backward linkability issue for verifier local revocation by introducing multiple intervals and corresponding revocation tokens. By taking small intervals, revealing revocation information only makes some transactions in the past (since the interval is small) and all future transactions linkable.

5.3.5 Analysis

Table 5.2 shows an overview of the aspects we discussed in Section 4.6 in as far they are relevant to group signatures. Not all alternative protocols have been included as in general they have worse properties than the scheme by Chen and Li [29]. The number of operations for the ACJT scheme is including the complicated proof of knowledge for revocation.

We note that revocation checking for the Chen and Li scheme only requires a single exponentiation with a fixed base element B. This means that using a special algorithm like [19] can speed-up the revocation checking even more. While the analysis of the Chen and Li scheme is somewhat superficial we can already see that it is a remarkably fast protocol. In Chapter 7 we provide a more detailed analysis of this protocol. This allows us to better compare it against our new protocol.

Scheme	ACJT [3]	BS [17]		CL [29]	
Private key (bits)	3072	513		769	
Signature size (bits)	large	1794		2308	
	ME	ME	PC	ME	PC
Signing	~ 500	8	3	6	
Verifying	~ 500	6	4+2$\|RL\|$	4+$\|RL\|$	1
Assumptions	sRSA, DDH	q-SDH, DLA		SDH, DDH	

Table 5.2: Aspects of revocation for group signatures. The abbreviation 'ME' denotes a (multi-)exponentiation, 'PC' denotes a pairing calculation and finally $|RL|$ is the lenght of the revocation list. Analysis adapted from [29].

5.4 Accumulators

The idea of an accumulator is to use a single value, the accumulate, to keep track of a whole list of values. Each accumulated value has a corresponding witness. Presenting this witness together with the value proves that the value is indeed accumulated. This principle can be used to add revocation to a credential scheme by accumulating the entire whitelist. A user then proves that he/she is on the whitelist by providing, or proving possession of, a witness. Of course, this approach does not yet provide any anonymity.

The concept of accumulators was first developed by Benaloh and de Mare in 1994 [10]. The construction guaranteed that it was hard to find a witness for a given item not on the list. This construction was made collision free by Barić and Pfitzmann in 1997 [4]. As a result of this extra property an attacker is not able to produce a witness for a not-accumulated value even if it can choose other accumulated values.

In 2002 Camenisch and Lysyanskaya [22] extended the scheme to allow dynamic additions and deletions of values in the accumulator. This accumulator was coined a dynamic accumulator. The amount of work required for these updates is linear in the number of changes.

5.4.1 Formal definition

Definition 5.3 (Secure Accumulator [20, 22])**.** A secure accumulator consists of a family of accumulation functions $\{\mathcal{F}_k\}$ indexed by the security parameter k. For every k a function $f \in \mathcal{F}_k$ is a function from $\mathcal{U}_f \times \mathcal{X}_k$ to $\mathcal{U}_f$, where $\mathcal{X}_k$ is the set of items that can be accumulated while $\mathcal{U}_f$ is the range of accumulator values for f. This family of functions should satisfy the following properties.

Efficient generation There is an probabilistic polynomial time algorithm AccGen that on input 1^k generates a random element $f \in \mathcal{F}_k$ together with some auxiliary information aux_f.

Efficient evaluation The function $f \in \mathcal{F}_k$ can be evaluated in polynomial time and $\mathcal{U}_f$ is efficiently samplable.

Quasi-commutative For all k, for all $f \in \mathcal{F}_k$, for all $u \in \mathcal{U}_f$, for all $x_1, x_2 \in \mathcal{X}_k$, $f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$. Hence, for $X = \{x_1, \ldots x_n\}$ it makes sense to define $f(u, X) := f(f(\ldots f(u, x_1), \ldots), x_n)$.

Witnesses Let $v \in \mathcal{U}_f$ and $x \in \mathcal{X}_k$. A value $w \in \mathcal{U}_f$ is called a witness for x in v under f if $v = f(w, x)$. Such a witness must always exist.

Security Let $\mathcal{U}'_f \times \mathcal{X}'_k$ be the domain for which the computational procedure for $f \in \mathcal{F}_k$ is defined. Then any probabilistic polynomial-time adversary A_k can only construct a witness w for an $x \notin X$ for some set X with negligible probability. Formally we say that the probability

$$Pr[f \leftarrow \mathsf{AccGen}; u \leftarrow \mathcal{U}_f; (x, w, X) \leftarrow A_k(f, \mathcal{U}_f, u) : X \subset \mathcal{X}_k \wedge w \in \mathcal{U}'_f \wedge x \in \mathcal{X}'_k \wedge x \notin X \wedge f(w, x) = f(u, X)],$$

where the $u \leftarrow \mathcal{U}_f$ denotes a sampling from the distribution $\mathcal{U}_f$ should be a negligible function of k. Note that only $X \subset \mathcal{X}_k$, x' may be chosen from $\mathcal{X}'$.

This definition can be extended to include dynamic accumulators. First note that every secure accumulator supports efficient addition of elements to the accumulator. Let $x \in X$ be a value that is accumulated into v, i.e. $v = f(u, X)$ and there exists a witness w for x such that $v = f(w, x)$. Upon adding $\tilde{x}$, all other witnesses need to be updated to correspond to the new accumulator value $v' = f(v, \tilde{x}) = f(u, X \cup \{\tilde{x}\})$. Then $w' = f(w, \tilde{x})$ is a witness for x in v', since $f(w', x) = f(f(w, \tilde{x}), x) = f(f(w, x), \tilde{x}) = f(v, \tilde{x}) = v'$.

The following definition states that for a dynamic accumulator there should be algorithms that can efficiently generate both a new accumulator value (algorithm D) as well as new witness for each remaining value (algorithm W) when a value is removed from the accumulator.

Definition 5.4 (Dynamic secure accumulator [20, 22])**.** A dynamic secure accumulator is a secure accumulator with the following additional property:

Efficient Deletion There exists efficient algorithms D and W such that, if $v = f(u, X)$, $x, \tilde{x} \in X$ and $f(w, x) = v$ then (1) $D(aux_f, v, \tilde{x}) = v'$ such that $v' = f(u, W \setminus \{\tilde{x}\})$; and (2) $W(w, v, v', x, \tilde{x}) = w'$ such that $f(w', x) = v'$.

Note that to generate a new accumulator value some auxiliary information aux_f about f might be needed.

5.4.2 RSA based constructions

The constructions suggested by Benahloh and de Mare [10], Barić and Pfitzmann [4], and Camenisch and Lysyanskaya [22] are all based on the concept of exponentiation modulo an RSA modulus n:

$$f(w, x) = w^x \pmod{n}.$$

Here $n = pq$ with $p = 2p' + 1, q = 2q' + 1, p', q'$ all primes. Note that f is indeed quasi-commutative and can be efficiently evaluated. The various methods differ in the restrictions placed on $\mathcal{U}_f$ and $\mathcal{X}_k$. In the remainder of this

section we will focus our attention to the construction proposed by Camenisch and Lysyanskaya [22].

Let $n = pq$ be an RSA modulus with p, q safe primes, i.e. they are of the above form. Furthermore, let A and B be two integers such that $A > 2$ and $B < A^2$. The auxiliary information aux_f consists of the factorization of n. The sets $\mathcal{X}$ and $\mathcal{U}_f$ are then defined as follows:

$$\begin{aligned}\mathcal{X} &= \{e \in \texttt{primes} : e \neq p', q' \wedge A \leq e \leq B\},\\ \mathcal{U}_f &= \{u \in QR(n) : u \neq 1\}.\end{aligned}$$

A value $\tilde{x}$ is added to the accumulator $v = f(u, X)$ by calculating $v' = f(v, \tilde{x}) = v^{\tilde{x}} \pmod n$. A witness u can be updated by calculating $u' = f(u, \tilde{x}) = u^{\tilde{x}} \pmod n$. To define how to removing a value $\tilde{x}$ from the accumulator we need to define algorithms D and W. The new accumulator value can be calculated as $v' = D((p, q), v, \tilde{x}) = v^{\tilde{x}^{-1} \pmod{(p-1)(q-1)}} \pmod n$.

Generating a new witness is a bit more involved. Let x be a value in the accumulator v and w its old witness, i.e. $v = f(w, x)$. Since x and $\tilde{x}$ are necessarily coprime we can find a, b such that $ax + b\tilde{x} = 1$. Then $w' = W(w, x, \tilde{x}, v, v') = u^b v'^a$, to verify this is correct we need to check that $v' = f(w', x) = (w')^x \pmod n$:

$$\begin{aligned}(w')^x &= (w^b (v')^a)^x\\ &= ((w^b (v')^a)^{x\tilde{x}})^{1/\tilde{x}}\\ &= ((w^x)^{b\tilde{x}} ((v')^{\tilde{x}})^{ax})^{1/\tilde{x}}\\ &= (v^{b\tilde{x}} v^{ax})^{1/\tilde{x}} = v^{1/\tilde{x}} = v'.\end{aligned}$$

The second equality sign holds because $\tilde{x}$ is coprime to $\phi(n)$. Note that the witness can be updated without knowledge of the factorization of n. In fact the same algorithms work when adding/removing multiple values by letting x' be the product of those values, see [22] for the details.

Under the hardness of the strong RSA problem, the above construction is a secure dynamic accumulator. The proof of this theorem can be found in Camenisch and Lysyanskaya [22]. The restriction on the range of allowable values is necessary to ensure security still holds when using batch updates. Without this restriction the proof by Barić and Pfitzmann [4] also works.

In addition to earlier approaches, Camenisch and Lysyanskaya [22] have also developed a zero-knowledge protocol for proving the possession of an accumulated value e in accumulator v. This protocol can be used to make an anonymous credential system based on whitelists. Since the protocol is zero-knowledge the prover does not reveal his witness nor the accumulated value itself.

The prover cannot directly reveal his witness and accumulated value in the proof, but it does need to show that the relation $u^e = v \pmod n$ holds. Therefore, the prover commits to a value e using a Pedersen commit $C_e = g^e h^r$ and a witness u using $C_u = uh^{r_2}$ and proves that $u^e = v \pmod n$ using these commited values.

Setup Generate two elements $g, h \in QR_n$ with n the RSA modulus and $\log_g h$ not known to the prover. The following values are common knowledge: C_e, g, h, n, v. The prover, in addition, knows u, r such that $u^e = v \pmod n$ and $C_e = g^e h^r$.

Protocol Phase 1: Prover chooses $r_1, r_2, r_3 \in_R \mathbb{Z}_{\lfloor n/4 \rfloor}$ and computes $C'_e = g^e h^{r_1}, C_u = uh^{r_2}, C_r = g^{r_2} h^{r_3}$ and sends C_e, C'_e, C_u and C_r to the verifier. Phase 2: Subsequently the prover and the verifier engage in the following zero-knowledge proof of knowledge:

$$PK\{(\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \phi, \psi, \eta, \sigma, \xi) :$$
$$C_e = g^\alpha h^\phi \wedge g = \left(\frac{C_e}{g}\right)^\gamma h^\psi \wedge g = (gC_e)^\sigma h^\xi \wedge C_r = h^\epsilon g^\zeta \wedge C'_e = h^\alpha g^\eta \wedge$$
$$v = C_u^\alpha (\frac{1}{h})^\beta \wedge 1 = C_r^\alpha (\frac{1}{h})^\delta (\frac{1}{g})^\beta \wedge \alpha \in [-B2^{k'+k''+2}, B2^{k'+k''+2}]\}$$

Details of the exact implementation of this protocol and a proof that it actually works can be found in [22]. In 2007 Li et al. [55] extended this protocol to include efficient zero knowledge proofs of non-inclusion.

5.4.3 Pairing based constructions

In 2009 Camenisch et al. [20] constructed a new dynamic accumulator that uses pairings. The main contribution is that as a result of using the pairings, witnesses can be more efficiently updated. Only a single multiplication is needed for each change. Furthermore these updates do not require any secret information, so can be delegated to an untrusted third party.

Just like the RSA version, the protocol employs zero-knowledge proofs of knowledge to demonstrate possession of a witness for an accumulated value. The proof of knowledge requires at least 18 exponentiations, and is thus slightly more expensive than the RSA version.

5.4.4 Analysis

The main advantage of using accumulators is that verification only takes constant time for both the user as well as the verifier. However there are also obvious downsides. Not only are the 18 exponentiations used in this proof considerably more than those used in for example the group-signature scheme, but it also needs an underlying credential system. Furthermore, every time the set of valid users changes the witnesses need to be updated. Especially in larger groups with more revocation this implies quite a lot of additional work. Camenisch et al. propose to use time slots of a single day to solve this, but it is not immediately obvious which advantages of accumulators remain as opposed to regular certificate systems with small expiration times.

5.5 Private sets

Consider two parties, Alice and Bob, each having private datasets respectively S_A and S_B, the cardinalities of which are publicly known. The problem of *private set intersection* is to let Alice and Bob compute the intersection $S_A \cap S_B$

such that only Alice learns the result and nothing more, while Bob learns nothing at all. Variants are *private set cardinality* testing where Alice only learns $|S_A \cap S_B|$ and *private set disjointness* testing where Alice only learns whether $S_A \cap S_B$ is empty or not. Some of the solutions we consider also support the more general solution where Bob also learns the answer that Alice learns.

These protocols can be used to to solve the revocation problem with whitelisting using the following approach. Let S_A equal $\mathcal{V}$, the set of unrevoked users. Then a user u can prove he/she still has access by setting $S_B = \{u\}$ and engaging with the verifier, who plays the role of Alice, in an private set intersection test. Note that then the set of users should be sparse to prevent a revoked user from simply posing as another unrevoked user. If the cardinality or disjointness test are used the identity of the user is not revealed to the verifier. Note that for singleton sets both tests are actually equal.

As a result of the generality of these protocols we cannot hope for a runtime that is below $O(|S_A + 1|)$ [38]. Furthermore, the protocols as described above can only be used for whitelisting and not for blacklisting as the user may just provide the protocol with an arbitrary value and hope it is not on the blacklist.

Several different solutions have been proposed to solve the private set intersection problem problem. The most common approach is to use oblivious polynomial evaluation. We examine the solution proposed by Freedman et al. [38] in more detail as it exposes most of the techniques that are generally used in the protocols. We briefly examine how Hohenberger and Weis improved the security of the protocol with respect to malicious participants [46]. Furthermore we see how Camenisch and Zaverucha [25] introduced certified sets to restricted the sets that are used in the protocol. This certificate can be used in the whitelist approach above. Each user then gets a certificate over the corresponding singleton set. Using an arbitrary set is then no longer possible. Finally we briefly see how Ye et al. [80] make a protocol that is robust against two malicious users.

5.5.1 Oblivious polynomial evaluation

Polynomials play an important role in solving the set intersection problem [38]. Let Alice have set $S_A = \{a_1, \ldots, a_{n_a}\}$ while Bob has set $S_B = \{b_1, \ldots, b_{n_b}\}$. Alice generates a polynomial $g(x) = \prod_{i=1}^{n_a}(x - a_i) = \sum_{i=0}^{n_a} \alpha_i x^i$, and sends the coefficients α_i to Bob. Bob then evaluates $c_i = g(b_i) + b_i$ for each $b_i \in S_B$ and sends the resulting c_i's back to Alice. Finally Alice outputs c_i if $c_i \in S_A$. Note that it is highly unlikely for c_i to be in S_A while c_i is not in S_B as long as the domain is sufficiently large.

The protocol in the previous paragraph does not, however, hide S_A from Bob (he can factor $g(x)$), nor does it hide S_B from Alice (she can solve $g(x)+x = c_i$). The latter can easily be solved by letting Bob return $c_i = r_i g(b_i) + b_i$ with r_i random instead. Note that c_i still equals b_i if $b_i \in S_A$. To solve the former Bob would need to evaluate the polynomial without actually knowing it and then let Alice recover the actual result. This problem is called oblivious polynomial evaluation and can be solved using homomorphic encryption.

Let (E, D) be a pair of instances of public-key encryption/decryption algorithms. Then this pair is called a homomorphic encryption scheme if

$$E(m_1)E(m_2) = E(m_1 + m_2) \quad \text{and} \quad E(m_1)^c = E(cm_1).$$

Suppose Bob is given $E(\alpha_0), \ldots, E(\alpha_{n_a})$ then he can calculate $E(g(x))$:

$$E(g(x)) = E(\sum_{i=0}^{n_a} \alpha_i x^i) = \prod_{i=0}^{n_a} E(\alpha_i)^{x^i}.$$

The actual responses $c_i = r_i g(b_i) + b_i$ are calculated similarly:

$$E(r_i g(b_i) + b_i) = \left(\prod_{i=0}^{n_a} E(\alpha_i)^{x^i} \right)^{r_i} E(b_i).$$

Several public key cryptosystems support homomorphic encryption. Freedman et al. used the Paillier public key crypto system [64], however this system suffers from larger ciphertexts. More recent schemes [25, 46] use the ElGamal public key crypto system [35] instead. The protocol we sketched in this section is essentially the original protocol designed by Freedman et al. [38].

If only the cardinality of the intersection is required then Bob can send $c_i = r_i g(b_i)$ instead. Alice only has to count the number of ciphertexts that decrypt to zero. This version would of course be the preferred one for revocation as it does not leak the users identity as long as Bob shuffles his responses.

5.5.2 Adding Security

The protocol suggested in the previous paragraph works fine as long as the participants are honest-but-curious. However in a setting of revocation we should not assume that Bob, the user, is honest. In fact he can easily fool the verifier into thinking he is on the whitelist be simply encrypting 0. Hohenberger and Weis [46] solved this by using a modified version of the ElGamal commitment scheme over a group G with composite order $n = pq$. The scheme is set up in such a way that only Alice can make valid encryptions and encrypting the zero value results in a point of order p. The hardness of the subgroup decision and subgroup computation problems ensure that it is hard to find such a point of order p.

Problem 5.5 (Subgroup Decision Problem (SDA))**.** Let G be a group of composite order $n = pq$ with $p < q$ k-bit primes. The the subgroup decision problem is to decide whether a point $x \in G$ has order p or not.

Problem 5.6 (Subgroup Computation Problem (SCA))**.** Let G be a group of composite order $n = pq$ with $p < q$ k-bit primes. The the subgroup computation problem is to find an $x \in G$ of order p.

Thus if the polynomial evaluation, which Bob can still do, results in a point of order p, Alice can be quite sure there is indeed an intersection.

An alternative construction was proposed by Ye et al. [80]. Their protocol is secure against malicious participants in the sense that even if both parties are

malicious then neither learns anything about the others input sets apart from the cardinality. The basis is again to represent the sets as polynomials, but this time also Bob computes such a polynomial. Then Alice and Bob cooperate in computing the Sylvester matrix S.

There are no intersections if and only if $\det(S) = 0$. By securely computing the determinant Alice learns whether there is an intersection. However she then also knows the cardinality of the intersection. This can be hidden by further blinding the Sylvester matrix. The protocol is set up in such a way that the components of the Sylvester matrix do not reveal the actual coefficients, but this is actually not a problem for white and blacklisting since then the cardinality of the intersection is at most one.

Note that this protocol offers no protection against Bob choosing an arbitrary dataset, however Bob cannot fake an intersection either. Camenisch and Zaverucha [25] use a scheme that is very similar to the original version proposed by Freedman et al., however they make the following additions:

- The sets S_A and S_B, and also the corresponding polynomials, are certified by a trusted third party. In case of a revocation application this can be used to ensure that a user can only use his true identity.
- Alice and Bob need to show that the sets they use are certified. This means that Alice must show her polynomial is certified, while Bob has to show that the b_i's he uses in the proof are certified.
- Finally Alice provides Bob with a verifiable decryption of the received values, in this way also Bob knows the cardinality.

The price of these additions are several zero-knowledge proofs that need to be provided by both Alice and Bob.

5.5.3 Analysis

While private set intersection protocols are very interesting there are too many problems when using them inside a credential system as they are. Only with the extensions by Camenisch and Zaverucha can we give the system as a whole protection against cheating users. Furthermore, all the protocols we have seen, have a linear dependency on the length of the input lists. This means that for each authentication we need to do work that is either linear in the number of users in the system or linear in the number of revoked users. Clearly, this is not desirable, especially not on weak devices like smart cards.

Chapter 6

Self-blindable credentials

In 1985 David Chaum already envisioned a credential system in which users are know to issuers only by their pseudonyms [26]. To show a credential from issuer A to issuer B the user would change, in some well-defined way, the pseudonym to that of issuer B before showing it to issuer B. Thus granting the user a weak form of anonymity. In the credential systems in the previous chapter, most notably Idemix and U-Prove, the necessity of this transformation is avoided by using zero-knowledge proofs. In 2001 Verheul [79] proposed a more straight-forward alternative that is closer to the original vision by Chaum. He built credentials that the user himself could transform from one pseudonym to another.

In Verheul's scheme there is no limit on the number of transformations, hence a user can even use a different pseudonym every time. Hence providing, at least intuitively, full anonymity since then the credentials cannot be linked anymore. This is why this kind of credentials is called self-blindable: the users can, by themselves, blind the credential. In this chapter we will first look at the definition of self-blindable credentials and then look at the resulting scheme by Verheul and the implementation for smart cards by Batina et al. In the next section we examine two protocols with self-blindable credentials with revocation. These two protocols claim to be unlinkable, but we shall see that this is in fact not the case.

6.1 Self-blindable credentials

A credential is self-blindable if any holder of a credential can transform it into a new credential that is then also accepted by the verifier. This notion is described formally in the following definition. It is a generalization from the definition in Verheul [79] to account for the other protocols that we will see.

Definition 6.1 (Self-blindable credentials (adapted from [79]))**.** Let $\mathcal{C}$ be the space of all possible credential objects, see Definition 4.1, and let $\mathcal{F}$ be the

space of private keys. Let $\mathsf{ShowCredential}(C, k)$ be the protocol that is run by the user to prove possession of a credential, represented by the credential object C with private key k, to a verifier. A credential is called *self-blindable* if there exists a blinding-factor space $\mathcal{B}$ and an efficiently computable *blinding map* $B : (\mathcal{C} \times \mathcal{F}) \times \mathcal{B} \to (\mathcal{C} \times \mathcal{F})$ with the following properties:

- Non-triviality: For any credential object $C \in \mathcal{C}$ and corresponding private key $k \in \mathcal{F}$ there exists $\alpha \in \mathcal{B}$ such that $B((C, k), \alpha) \neq (C, k)$.
- Validity: For any $\alpha \in \mathcal{B}$, any credential object $C \in \mathcal{C}$ and corresponding private key $k \in \mathcal{F}$ if running $\mathsf{Authenticate}(C, k)$ with V results in an accept, then $\mathsf{Authenticate}(B((C, k), \alpha))$ also results in an accept by V.

The non-triviality property ensures that the blinding map is non-trivial. The second property ensures that a blinded certificate can also be used to demonstrate possession of a credential.

Hereafter we will consider first the scheme proposed by Verheul and then an adaptation for smart cards. We always consider a single issuer that issues a single type of credential. Extensions to multiple issuers or multiple types of credentials are straightforward.

6.1.1 Verheul

The self-blindable credentials proposed by Verheul are just certificates over a public key just as with traditional systems. The certificate is a Chaum and Pedersen signatures as described in Section 3.6.1. The simple form of this signature scheme allows these certificates to be self-blindable.

We first fix some notation for the remainder of this chapter. Let (G_1, G_2) be a bilinear group pair and $e : G_1 \times G_2 \to G_3$ be the corresponding pairing. Let G_1, G_2, and G_T be of prime order p, with P be a generator of G_1, and Q a generator of G_2. In addition we have a set of user with private keys k_i and corresponding public keys $K_i = k_i P$.

Let $A = aQ$ be the public key of the issuer. Recall that a Chaum-Pedersen signature over a public key K_i is then given by $C_i = aK_i$. Such a signature can hence checked by a verifier using the pairing check

$$e(C_i, Q) = e(K_i, A). \tag{6.1}$$

For this scheme Verheul uses a super-singular curve, so the pairing is a Type 1 pairing. The credential object is given by the tuple (K_i, C_i). In Section 3.6.1 we mentioned that this signature scheme is not very secure as it allows existential forgeries: given a certificate C_i over a key K_i the point αC_i is a certificate over αK_i for any $\alpha \in \mathbb{F}_p$. Verheul realized that this precise weakness can be used to make this scheme self-blindable. In fact, the blinding-factor space $\mathcal{B}$ is $\mathbb{F}_p^*$ while the blinding map B is given by

$$B((C, X, x), \alpha) = (\alpha C, \alpha X, \alpha x).$$

Let $(\overline{C_i}, \overline{K_i}, \overline{k_i}) = B((C_i, K_i, k_i), \alpha)$, then clearly equation (6.1) is also satisfied when replacing C with $\overline{C_i}$ and K_i with $\overline{K_i}$.

User		*Verifier*
blinding $\alpha \in_R \mathbb{Z}_p$		
send αK_i , αC_i	$\longrightarrow$	**into** $\overline{K}$, $\overline{C}$
		verify $e(\overline{K}, A) \stackrel{?}{=} e(\overline{C}, Q)$
		nonce $\eta \in_R \mathbb{Z}_p$
into η'	$\longleftarrow$	**send** η
send $(r, s) = \mathsf{signECDSA}(\eta', \alpha k_i)$	$\longrightarrow$	**into** (r', s')
		$\mathsf{verifyECDSA}(r', s', \overline{K})$

Figure 6.1: Basic protocol using self-blindable credentials by Batina et al. [5]. It combines the verification part proposed by Verheul with a challenge-response protocol to ensure the user knows the private key.

Verheul correctly emphasizes that $k_i = 0$, or equivalently $K_i = C_i = O$ should *not* be allowed. Equation 6.1 is then trivially satisfied, but the security is in no way guaranteed. A practical implementation of Verheul's system is shown in the next section.

6.1.2 Batina et al.

Batina et al. [5] implemented the credentials by Verheul on a smart card. Their first attempt is shown in Figure 6.1. The first part is as could be expected from the description of the protocol above. However, as we saw earlier, we also need to guard against replay attacks. That is why the check of the credential is followed by a challenge-response part where the user signs a random nonce generated by the verifier using the private key αk_i corresponding to the blinded public key αK_i. The signature algorithm used here is the elliptic curve variant of DSA, that is implemented on the smart card.

Note that one has to be careful if there exists an efficiently computable homomorphism $\phi : G_2 \to G_1$ such that $P = \phi(Q)$, then an attacker can pick a random key k_i and compute $K_i = \phi(k_i Q)$ and $C_i = \phi(k_i A)$ (using the public values Q and A for an arbitrary attribute of his choosing) to create a credential that will pass all tests[1]. However, as long as such a homomorphism is not efficiently computable this trick is not possible.

The second protocol by Hoepman et al. [45] is presented in Fig. 6.2, which uses a simple point multiplication and corresponding check in the challenge-response exchange. This improves the performance by a factor of 2 when implementing the prover side on a smart card [45]. Note that in principle the order of committing to the blinding factor and sending the nonce could be interchanged to achieve a two-message protocol (by combing the commitment to the blinding with the 'signing' of the nonce).

The trick where the nonce N is signed by multiplying it with the private key αk_i is actually just a Chaum-Pedersen signature over N corresponding to the

[1] This omission was pointed out by Christian Wachsmann.

Prover		*Verifier*
blinding $\alpha \in_R \mathbb{Z}_p$		
send αK_i , αC_i	$\longrightarrow$	**into** $\overline{K}$, $\overline{C}$
		verify $e(\overline{K}, A) \stackrel{?}{=} e(\overline{C}, Q)$
		nonce $\eta \in_R \mathbb{Z}_p$
into N	$\longleftarrow$	**send** ηP
send $\alpha k_i N$	$\longrightarrow$	**into** $\overline{M}$
		verify $\overline{M} \stackrel{?}{=} \eta\overline{K}$

Figure 6.2: Faster protocol using self-blindable credentials by Hoepman et al. [45]. Speedup obtained by not using the ECDSA algorithm for signing the nonce but using the simpler Chaum-Pedersen style signature instead.

public key αK_i. In this special case the discrete logarithm of N w.r.t. P is known by the verifier, therefore it can use a very fast multiplication check to verify the signature.

Both protocols are vulnerable to compromises of the smart card. Should the private key k_i and the certificate C_i leak, many clones of the card (with different private key xk_i and certificate xC_i) are easily made, although this does not involve attributes for which the original card did not own the corresponding credentials. In fact, by taking $x = k(1/k_i)$ the user can make certificates for any private key k. The Kiyomoto protocol [52] discussed further in Sect. 6.2.1 has the same drawback.

6.2 Adding revocation

The problem with the protocols we saw in the previous section is that they cannot, in any way, be used for revocation. Precisely because the user can trivially make a new certificate over a new public key there is nothing that binds a credential to a user. From an anonymity perspective this is perfect, but it also means that revocation is impossible.

In 2008 Kiyomoto and Tanaka [52] proposed the first self-blindable credential scheme with revocation. We shall see that this scheme, which still uses the Chaum-Pedersen signatures is incorrect. Later Emura et al. [36] came up with a better protocol. The important innovation in the latter is the use of Boneh and Boyen signatures, see Section 3.6.2 instead of the Chaum-Pedersen signatures. We first examine the Kiyomoto scheme and its deficiencies, after which we examine the scheme by Emura and show that this scheme is also not completely correct.

6.2.1 Kiyomoto

The Kiyomoto et al. [52] protocol (using our notation where $A = aQ, K_i = k_i P$ and $C_i = aK_i$) is presented in Fig. 6.3. Again we have the issuers public key $A = aQ$, the public key of the user $K_i = k_i P$ and a certificate $C_i = aK_i$. In this protocol the private key of a user i equals $k_i = \kappa_i + \kappa'_i$ with κ_i random,

Prover		*Verifier*
blinding $\alpha \in_R \mathbb{Z}_p$		
send αK_i , αC_i	$\longrightarrow$	**into** $\overline{K}$, $\overline{C}$
		verify $e(\overline{K}, A) \stackrel{?}{=} e(\overline{C}, Q)$.
		nonce $\eta \in_R \mathbb{Z}_p$
into N	$\longleftarrow$	**send** ηP
send $\alpha\kappa_i N$, $\alpha\kappa_i' N$	$\longrightarrow$	**into** $\overline{M}$, $\overline{M}'$
		verify $e(\overline{M} + \overline{M}', P) \stackrel{?}{=} e(\overline{K}, \eta P)$
		for every entry (R, S)
		in the revocation list:
		verify $e(S, \overline{M}) \stackrel{?}{=} e(\overline{K}, \eta R)$

Figure 6.3: Revocation protocol of Kiyomoto et al. [52]

and $\kappa_i' = m_i\kappa_i$ a *non-repudiation private key*, where m_i is a number encoding some valuable piece of information related to the identity of i. If both κ_i and κ_i' become known, then so does m_i, by a simple division. Hence, so the authors argue, this discourages a user from sharing both κ_i as well as κ_i'. The revocation list contains pairs $(\kappa_j P, C_j = ak_j P)$ for revoked users j. The blinding-factor space $\mathcal{B}$ is again $\mathbb{F}_p^*$ and the blinding map B is given by

$$B((C_i, K_i, \kappa_i, \kappa_i'), \alpha) = (\alpha C_i, \alpha K_i, \alpha\kappa_i, \alpha\kappa_i'). \tag{6.2}$$

In the original protocol the value aP was used as a public key for the issuer. This immediately allows any user to create his certificates just by multiplying with a suitable private key. This problem is fixed here because only aQ is known, and we again assume that no homomorphism mapping Q to P is known. This problem was pointed out by Emura et al. [36]. Unfortunately, two shortcomings remain. First of all, the constraint $\kappa_i' = m_i\kappa_i$ is not enforced by the protocol. Thus, if a revoked user knows its private key k_i, it can generate new keys λ_i, λ_i' with $\lambda_i + \lambda_i' = k_i$ while $\lambda_i \neq \kappa_i$. It is easily checked that the first condition guarantees that the certificate is still valid, while the second condition ensures that the keys are no longer recognized as being revoked.

Secondly, a malicious verifier (that controls the challenge η) can trace a user j after it has intercepted one protocol run of user j. To this end it computes a valid revocation list entry $(\kappa_j P', k_j P')$ by storing the first part of the third message $(\alpha\kappa_j)(\eta P)$ while multiplying the first part of the first message αK_j (which equals $\alpha k_j P$) by η. Then $P' = \eta\alpha P$, and the revocation check returns true for user i. Note that this revocation check is on a per-user basis, it is not possible to revoke only a single credential.

So we conclude that this protocol is neither secure, nor anonymous.

6.2.2 Emura et al.

A simplified version of the Emura et al. [36] protocol is shown in Fig. 6.4. The original version additionally contains an authentication of the verifier. The

important idea is to use Boneh and Boyen signatures, see Section 3.6.2 to prevent the common cloning attacks we saw in various forms in the previous sections. Now the certificate is $C_i = \frac{1}{k_i+a}P$, a signature by $A = aQ$ over the key k_i. Normally, such a signature would be verified using equation

$$e(C_i, K_i + A) = e(P, Q),$$

where $K_i = k_iQ$, as we saw in Section 3.6.2 (although the roles of G_1 and G_2 are now interchanged). From an anonymity point of view this is not good enough as both C_i as well as K_i are unique for the user.

Let us try to derive heuristically what this equation should look like if we try to apply the blinding trick and set $\overline{C_i} = \alpha C_i$ and $\overline{K_i} = \alpha K_i$. We immediately run into trouble: the left hand side does not evaluate to something useful. This can be improved by sending $\overline{A} = \alpha A$ as well, then $e(\overline{C_i}, \overline{A} + \overline{K_i}) = e(P, Q)^{\alpha^2}$. To complete the verification we need two factors α on the right as well. We can do this by including $\overline{P} = \alpha P$ and $\overline{Q} = \alpha Q$. The verification equation then becomes:

$$e(\overline{C_i}, \overline{A} + \overline{K_i}) = e(\overline{P}, \overline{Q}),$$

which is quite close to the equation Emura et al. came up with. The main difference is that instead of using $\overline{K_i}$ they use $\overline{M} = \eta\overline{K_i}$ thus the verification of the Chaum-Pedersen signature is included in the verification of the signature:

$$e(\overline{C_i}, \eta\overline{A} + \overline{M}) = e(\overline{P}, \overline{N}). \tag{6.3}$$

However, this equation alone is not enough to guarantee that $\overline{C_i}$ is really a good signature. For example by setting $\overline{M} = xN + X$ and $\overline{A} = -xQ$, the second argument reduces to just X. By setting $\overline{C_i} = \overline{P}$ and $\overline{N} = X$ the equation is trivially satisfied. To make sure this does not happen the equation

$$e(\overline{P}, A) = e(P, \overline{A})$$

ensures that $\overline{A}$ is really equal to A times the blind. Similarly,

$$e(\overline{P}, \eta Q) = e(P, \overline{N})$$

ensures that $\overline{N}$ is really N multiplied by the blind. This heuristic argument explains why the checks are likely to be correct. Unfortunately, the proof given by Emura et al. has some issues that we do not know how to fix. Just as with the Kiyomoto protocol we can write down the blinding map for a transformation factor space $\mathbb{F}_p^*$

$$B((C_i, A, 1, k_i), \alpha) = (\alpha C_i, \alpha A, \alpha, \alpha k_i).$$

Note that we need a couple of extra values in the credential object to make sure we have enough information to complete the ShowCredential protocol.

To revoke a credential for user i, an entry is added to the revocation list of the form $(\gamma C_i, \gamma P)$, where $\gamma \neq 1$ is random. Intuitively, the presence of γP allows the verifier to take out the blind in γC_i during the revocation check, but does not allow the verifier access to a valid credential C_i. The verifier can check whether a certificate is on the list by plugging this blinded certificate into

Prover		*Verifier*
		nonce $\eta \in_R \mathbb{Z}_p$
into N	$\longleftarrow$	**send** ηQ
blinding $\alpha \in_R \mathbb{Z}_p$		
send $\alpha C_i\,,\ \alpha A\,,\ \alpha k_i N\,,\ \alpha N\,,\ \alpha P$	$\longrightarrow$	**into** $\overline{C}\,,\ \overline{A}\,,\ \overline{M}\,,\ \overline{N}\,,\ \overline{P}$
		verify $e(\overline{P}, A) \stackrel{?}{=} e(P, \overline{A})$.
		verify $e(\overline{P}, \eta Q) \stackrel{?}{=} e(P, \overline{N})$.
		verify $e(\overline{C}, \eta\overline{A} + \overline{M}) \stackrel{?}{=} e(\overline{P}, \overline{N})$.
		for every entry (C, R) in the revocation list verify that: $e(C, \eta\overline{A} + \overline{M}) \stackrel{?}{\neq} e(R, \overline{N})$.

Figure 6.4: Revocation protocol of Emura et al. [36] (simplified).

equation (6.3) and modifying the right hand side accordingly. If this equation is also satisfied then the certificate was on the revocation list.

The above heuristic argument is very dangerous indeed. In fact, this protocol is not correct in the sense that it does not satisfy the anonymity requirements, even though all transmitted values are blinded. We do believe the heuristic arguments given above with regards to unforgeability are correct, but we cannot prove this. As it turns out, transactions are almost trivially linkable because the values $\overline{C}, \overline{P}$ exactly constitute a revocation entry, and are transmitted by the user.

Chapter 7

Provable secure protocol

In this chapter we present our new self-blindable credential protocol that is provably secure. More concretely, we solve the problem of linkability in the Emura protocol and we provide a formal proof for both unforgeability as well as anonymity. Just like the protocol by Emura et al. [36] we use Boneh and Boyen signatures[1], see Section 3.6.2 as the basis for our scheme. We made the following significant changes to the protocol by Emura et al. [36]:

- we use two different blinding factors to ensure that the pairing trick that made Emura et al.'s protocol linkable cannot be used in our protocol;
- we use the general version of the Boneh and Boyen signature scheme to allow for a formal proof of unlinkability;
- we replaced the signatures on random nonces by zero-knowledge proofs of knowledge to aid the proof of unforgeability of the protocol; and
- we use a different revocation check that is significantly faster to perform.

The structure of this chapter is as follows. In the next section we first informally motivate the design decision in our protocol. Then we give a description of the full protocol. In Section 7.3 we give formal definitions of the security games defining the security of our protocol. Next, in Section 7.4 we give the proofs of the security of our protocol. Finally, in Section 7.5 we compare our method with other self-blindable credential protocols.

In the next chapter we heuristically transform the protocol to versions that are more practical at the cost of weakening the provable security. In addition, we shall extend the protocol we propose here to a full credential system with multiple issuers and credentials.

7.1 Motivation

We first examine the exact nature of the linkability problem in the Emura protocol. Suppose an adversary obtains the values $\overline{P}_1 = \alpha_1 P$ and $\overline{C}_1 = \alpha_1 C_1 =$

[1] We thank Pim Vullers for suggesting this approach.

$\alpha_1 c_1 Q$ from one interaction and $\overline{P}_2 = \alpha_2 P$ and $\overline{C}_2 = \alpha_2 C_2 = \alpha_2 c_2 Q$ from another interaction. Recall that C_i represents a certificate which is unique for a given user. Now, an adversary can check the validity of the following equation

$$e(\overline{P}_1, \overline{C}_2) \stackrel{?}{=} e(\overline{P}_2, \overline{C}_1)$$

to determine whether C_1 equals C_2. To see this, note that

$$e(\overline{P}_1, \overline{C}_2) = e(\alpha_1 P, \alpha_2 c_2 Q) = e(P, Q)^{\alpha_1 \alpha_2 c_2}$$
$$e(\overline{P}_1, \overline{C}_2) = e(\alpha_2 P, \alpha_1 c_1 Q) = e(P, Q)^{\alpha_1 \alpha_2 c_1},$$

so the former equality holds exactly when $c_1 = c_2$. The pairing allows us to cancel the blinds, and hence solve the underlying cross-group Diffie-Hellman problem. This led to the first idea that guided the construction of our protocol: ensure that the cross-group Diffie-Hellman problem is not usable by using different blinds for G_1 and G_2. This prevents the above trick from working.

Now consider αP and $\alpha k P$, i.e. both in G_1. One might wonder whether this gives any problems. Indeed it does: if a type 1 pairing exists for G_1, see Section 3.5, the above trick (now involving $\overline{P} = \alpha P$ and $\overline{K} = \alpha k P$) solves the ordinary Decisional Diffie-Hellman problem in G_1 and thus gives linkability even if both points are in G_1. So type 1 pairings cannot be used. Now consider αQ and $\alpha c Q$ both in G_2. By applying the homomorphism ϕ from G_2 to G_1 from a type 2 pairing the trick can again be made to work and used to solve the Decisional Diffie-Hellman problem in G_2. We conclude, just like Emura et al., that the Decisional Diffie-Hellman problem should be hard in both G_1 and G_2. Hence, we have to use a type 3 curve.

We deviate significantly from Emura et al. in the revocation method we use. We use the idea for revocation used in Chen and Li [29], see Section 5.3. Suppose a user provides both $\overline{P} = \alpha P$ as well as $\overline{K} = \alpha k P$. Then for a revoked private key x on the revocation list a verifier can easily check whether they match the user by testing whether

$$x\overline{P} \stackrel{?}{=} \overline{K}.$$

Thus, the calculation of two pairings can be replaced by a single point-multiplication. The latter is at least an order of magnitude faster than a pairing check. Unfortunately, our scheme also inherits a property from group signatures that is less desirable for our scheme: given the revocation token also past transaction become linkable.

7.2 Protocol description

Our self-blindable credential protocol consists of a number of subprotocols. The algorithm Setup creates a new system and provides the users of the system with credentials. In practice this can be a distributed process where one party sets the system parameters and various authorities provide the attributes. Furthermore, user can be added dynamically instead of statically as we do now. To simplify the description of the system we limit ourselves to only one possible attribute. A user can prove the possession of an attribute by executing the

interactive ShowCredential protocol with a verifier. This protocol consists of two sides: Prove and Verify. The prover runs Prove and sends the results to the verifier. The verifier checks this proof using Verify. This separation makes it easier to formally define correctness of the system. Finally the protocol Revoke can be used to revoke a user.

Definition 7.1 (SBR protocol)**.** The following protocols describe our new self-blindable credential scheme with revocation. In the following protocol $h : \{0,1\}^* \to \mathbb{F}_p$ is a hash-function that is treated as a random oracle in the security proofs.

Setup$(1^k, n)$**:** The algorithm Setup is run by the certificate authority. First, it runs instance generator $I(1^k)$ to get a bilinear group pair (G_1, G_2) with security parameter k and a pairing function $e : G_1 \times G_2 \to G_T$. Let p be the order of G_1 and G_2. Next, the algorithm generates two generators $P \in G_1$ and $Q \in G_2$. The certificate authority generates two random private keys $a, b \in_R \mathbb{Z}_p$ and sets the corresponding public keys $A_P = aP, A_Q = aQ, B = bP$ accordingly. For each of the n users it generates a private key $k_i \in_R \mathbb{Z}_p$ and a random number $r_i \in_R \mathbb{Z}_p$, such that $k_i + a + r_i b \in \mathbb{F}_p^*$. It uses these values to calculate the certificates

$$C_i = \frac{1}{k_i + a + r_i b} Q.$$

The certificate authority makes the values P, Q, A_P, A_Q, B public. Furthermore, it sends to each user i its private values (k_i, r_i, C_i). Finally, it sets the revocation list RL, consisting of revoked private keys, to $\emptyset$.

Prove(k_i, C_i, r_i, η)**:** In response to a nonce η from the verifier the prover i generates two blinds $\alpha, \beta \in_R \mathbb{Z}_p$ and creates the following set of blinded values:

$$\begin{aligned} &\overline{K} = \alpha K_i, \quad \overline{A} = \alpha A_P, \quad \overline{B} = \alpha r_i B, \quad \overline{P} = \alpha P, \quad (\in G_1) \\ &\overline{Q} = \beta Q, \quad \overline{C} = \beta C_i \quad (\in G_2), \end{aligned}$$

where $K_i = k_i P$. Furthermore, the prover creates a signature over the nonce η using the following signature proof of knowledge:

$$\begin{aligned} \sigma = SPK\{(\alpha, \alpha k_i, \alpha r_i, \beta) : \overline{P} = \alpha P \ \wedge\ \overline{K} = (\alpha k_i)P \ \wedge \\ \overline{B} = (\alpha r_i)B \ \wedge\ \overline{Q} = \beta Q\}(\eta). \end{aligned}$$

In practice this signature will take the following form. First the prover generates ephemeral keys $r_\alpha, r_k, r_r, r_\beta \in_R \mathbb{F}_p$ and creates the commitments $R_\alpha = r_\alpha P, R_k = r_k P, R_r = r_r B, R_\beta = r_\beta Q$. Then it calculates the challenge $e = h(R_\alpha \parallel R_k \parallel R_r \parallel R_\beta \parallel \eta)$. Finally the prover calculates the responses $s_\alpha = r_\alpha + \alpha e, s_k = r_k + (\alpha k_i)e, s_\beta = r_\beta + \beta e, s_r = r_r + (\alpha r_i)e$. The signature σ over the nonce η is then given by the tuple $\sigma = (e, s_\alpha, s_k, s_r, s_\beta)$. The prover then sends the response $\xi = (\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma)$ to the verifier.

Verify(η, ξ, RL)**:** Let the response ξ equal $(\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma)$. The verifier verifies the signature proof of knowledge σ. It ensures that none of the

received points equal O and checks that the following equations hold:

$$e(\overline{A}, Q) \stackrel{?}{=} e(\overline{P}, A_Q) \tag{7.1}$$

$$e(\overline{K} + \overline{A} + \overline{B}, \overline{C}) \stackrel{?}{=} e(\overline{P}, \overline{Q}) \tag{7.2}$$

The verifier performs the revocation check by ensuring that for none of the keys $x \in RL$ the equation $x\overline{P} = \overline{K}$ holds. If all of the above checks pass the verifier returns `valid`, otherwise it returns `invalid`.

ShowCredential(): The authenticate protocol is an interactive protocol between the prover and the verifier. The prover requests to authenticate with the verifier. The verifier replies by generating a random nonce $\eta \in \mathbb{F}_p$ and sending it to the prover. The prover executes $\xi = \text{Prove}(k_i, r_i, C_i, \eta)$ with his private key k_i, and certificate (C_i, r_i). It sends the response ξ to the verifier. The verifier checks this response using $\text{Verify}(\eta, \xi, RL)$.

Revoke(k): The certificate authority can revoke a user by adding the users' private key k to the revocation list RL.

Figure 7.1 contains a graphical representation of the ShowCredential protocol between a prover and a verifier. The signature proof of knowledge has been expanded to allow for easier complexity analysis later on.

We opted to use a signature obtained by the Fiat-Shamir heuristic on a nonce instead of an interactive zero-knowledge proof for two reasons. First, it allows for a simple two-pass protocol that is easier to analyze. Second, it produces a protocol that is conceptually easier to transform to more practical approaches as we shall see in the next chapter. The results remain valid when the signature is replaced by an interactive zero-knowledge proof. The random oracle model is no longer needed in this case. In either case, the structure of the zero-knowledge proof allows us to fake signatures in our proofs, either by rewinding an honest verifier, or by modifying the random oracle. Furthermore, the structure also allows us to extract knowledge from a cheating forger by rewinding a prover.

7.3 Games and definitions

As our system is more specific then our formal model from Chapter 4 we need to give a new definition of security of the system. We do this by defining correctness of a scheme and by giving two security games: one for unforgeability and one for anonymity.

Correctness of the system means that any unrevoked user with a valid credential is accepted by any verifier.

Definition 7.2 (Correctness)**.** For any set of private values (k_i, r_i, C_i) of a user generated by Setup, any nonce $\eta \in \mathbb{F}_p$ and any response $\xi = \text{Prove}(k_i, C_i, r_i, \eta)$ generated by the prover the following holds:

$$\text{Verify}(\eta, \xi, RL) = \texttt{valid} \quad \Leftrightarrow \quad k_i \notin RL. \tag{7.3}$$

Next, we define the unforgeability game. The goal of the adversary is to produce a forgery of some credential, i.e. succeed in producing al response ξ to

Prover		*Verifier*
		nonce $\eta \in_R \mathbb{Z}_p$
into η	$\longleftarrow$	**send** η
blinding $\alpha, \beta \in_R \mathbb{Z}_p$		
send $\alpha K_i, \alpha A_P, \alpha r_i B$	$\longrightarrow$	**into** $\overline{K}, \overline{A}, \overline{B}$
send $\alpha P, \beta Q, \beta C_i$	$\longrightarrow$	**into** $\overline{P}, \overline{Q}, \overline{C}$
		verify:
		$e(\overline{A}, Q) \stackrel{?}{=} e(\overline{P}, A_Q)$.
		$e(\overline{K} + \overline{A} + \overline{B}, \overline{C}) \stackrel{?}{=} e(\overline{P}, \overline{Q})$.
generate $r_\alpha, r_k, r_r, r_\beta \in_R \mathbb{Z}_p$		
calculate $R_\alpha = r_\alpha P, R_k = r_k P$		
calculate $R_r = r_r B, R_\beta = r_\beta Q$		
let $e = h(R_\alpha \parallel R_k \parallel R_r \parallel R_\beta \parallel \eta)$		
send $e,\ r_\alpha + \alpha e,\ r_k + (\alpha k_i)e$	$\longrightarrow$	**into** $e',\ s_\alpha,\ s_k$
send $r_\beta + \beta e,\ r_r + (\alpha r_i)e$	$\longrightarrow$	**into** $s_\beta,\ s_r$
		verify:
		$R'_\alpha \stackrel{?}{=} s_\alpha P - e'\overline{P}$.
		$R'_k \stackrel{?}{=} s_k P - e'\overline{K}$.
		$R'_r \stackrel{?}{=} s_r B - e'\overline{B}$.
		$R'_\beta \stackrel{?}{=} s_\beta Q - e'\overline{Q}$.
		$e' \stackrel{?}{=} h(R'_\alpha \parallel R'_k \parallel R'_r \parallel R'_\beta \parallel \eta)$
		for every entry k
		in the revocation list:
		verify $k\overline{P} \stackrel{?}{\neq} \overline{K}$.

Figure 7.1: Complete ShowCredential protocol as described in Definition 7.1 between a prover and a verifier. The protocol proceeds in three phases: first the prover sends the blinded keys and certificates which are checked by the verifier, then the prover sends a non-interactive proof of knowledge for the private values which is checked by the verifier. Finally, the verifier checks whether the prover's private key was revoked.

a nonce η that is accepted by Verify. The adversary could pretend to be a verifier, hence we allow the adversary to interact with any user following the ShowCredential protocol. Furthermore, we allow the adversary to corrupt any user. The adversary's goal is then to produce a valid response that corresponds to an unrevoked and uncompromised user.

Definition 7.3 (Unforgeability)**.** The unforgeability game between a challenger and a adverary $\mathcal{A}$ is defined as follows.

Setup$(1^k, n)$ The challenger runs the Setup$(1^k, n)$ algorithm, obtaining a bilinear group pair (G_1, G_2), a pairing e, public points P, Q, A_P, A_Q, B and

private keys a, b. The challenger stores the private information for the n users. Furthermore, it provides the adversary with the description of the groups and the pairing, as well as the public points P, Q, A_P, A_Q, B and sets $\mathcal{C}$, the adversary's coalition, to $\emptyset$.

Hash Queries Algorithm $\mathcal{A}$ can make hash queries for any string. These queries are answered by the challenger.

Queries Algorithm $\mathcal{A}$ can make queries of the challenger, as follows.

Prove(i, η) The adversary $\mathcal{A}$ can request the output of the Prove protocol for user i and random nonce η of its choice. The challenger runs $\mathsf{Prove}(\eta, k_i, r_i, C_i)$ and returns the output $(\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma)$ to the adversary.

Corrupt(i) The adversary $\mathcal{A}$ can request the private key and certificate of user i. The challenger looks up user i and responds with (k_i, r_i, C_i). It also adds k_i to $\mathcal{C}$.

Revoke(i) The adversary $\mathcal{A}$ can request the revocation token of user i. The challenger looks up user i and responds with k_i, the revocation token. It also adds k_i to $\mathcal{C}$.

Challenge The challenger sends a nonce η to the adversary.

Response The adversary outputs a response $\xi = (\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma)$ to the nonce η.

The adversary $\mathcal{A}$ wins the unforgeability game if its output is accepted by $\mathsf{Verify}(\eta, \xi, \mathcal{C})$. Note that an explicit query to a verifier to perform the verification is not necessary since the adversary can do this without help.

We define the probability that $\mathcal{A}$ wins this game as $\mathsf{AdvU}_{\mathcal{A}}$, the probability is over the coin tosses of $\mathcal{A}$ and the randomized generation of the keys and signing algorithms.

Definition 7.4. A forger $\mathcal{A}$ $(t, n, q_V, q_h, \epsilon)$-breaks unforgeability of the n-user self-blindable credential scheme if $\mathcal{A}$ runs in time at most t, makes at most q_P prove requests and q_h hash queries, and $\mathsf{AdvU}_{\mathcal{A}}$ is at least ϵ.

Finally, we define the anonymity game. An adversary wins this game by distinguishing between the responses of two users of its own choosing. Note that this is very similar to the notion of indistinguishability of encryptions in Section 2.2.1.

Definition 7.5 (Anonymity)**.** The anonymity game between a challenger and an adversary $\mathcal{A}$ is defined as follows.

Setup As in the unforgeability game.

Hash Queries As in the unforgeability game.

Queries Algorithm $\mathcal{A}$ can make queries of the challenger, as follows.

Prove(i, η) As in the unforgeability game.

Corrupt(i) As in the unforgeability game.

Revoke(i) As in the unforgeability game.

Challenge The adversary outputs a nonce η and the indices i_0, i_1 of two members, such that $\mathcal{A}$ neither corrupted nor revoked users i_0 nor i_1, i.e. $i_0, i_1 \notin \mathcal{C}$. The challenger chooses a bit $d \in \{0,1\}$ and runs $\mathsf{Prove}(\eta, k_{i_d}, r_{i_d}, C_{i_d})$ for user i_d to obtain a valid response $(\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma)$ to nonce η. It sends this response to the adversary.

Restricted Queries After receiving the challenge $\mathcal{A}$ can make additional queries of the challenger, restricted as follows.

- **Prove**(i, η) As before.
- **Corrupt**(i) As before, but $\mathcal{A}$ cannot make corruption queries for users i_0 and i_1.
- **Revoke**(i) As before, but $\mathcal{A}$ cannot make revocation queries for users i_0 and i_1.

Output Finally, $\mathcal{A}$ outputs a bit d', its guess of d. The adversary wins if $d = d'$.

Note that if an adversary can break unlinkability it can also win this game. We define $\mathcal{A}$'s advantage in winning this game as $\mathsf{AdvA}_{\mathcal{A}} := |\Pr[d = d'] - 1/2|$, where the probability is over the coin tosses of $\mathcal{A}$, the randomized generation of the keys and signing algorithms, and the choice of b. We note that any deviation from the average $1/2$ is an improvement, hence explaining the absolute values.

Definition 7.6. An adversary $\mathcal{A}$ $(t, n, q_V, q_h, \epsilon)$-breaks the anonymity of an n-user system if $\mathcal{A}$ runs in time at most t, makes at most q_P prove requests and q_h hash queries, and $\mathsf{AdvA}_{\mathcal{A}}$ is at least ϵ.

7.4 Proofs of security

We show the anonymity and unforgeability of our self-blindable credential protocol. The structure of the proofs is inspired by the proofs for selfless-anonymity and traceability, two related concepts, for a group signature scheme by Boneh and Shacham [17].

Theorem 7.7. *The self-blindable credential scheme is correct, as defined in definition 7.2.*

Proof. Consider a prover with private key k_i and signature (C_i, r_i), so $e(k_i P + r_i B + A, C_i) = e(P, Q)$. Let $\eta \in \mathbb{Z}_p$ be a random nonce and $\xi = \langle \overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma \rangle = \text{Prove}(k_i, C_i, r_i, \eta)$ the response by the prover. Let $\alpha, \beta \in \mathbb{Z}_p$ be the blinds used by the prover, i.e. $\overline{P} = \alpha P$ and $\overline{Q} = \beta Q$. Then equation (7.1) is satisfied because

$$e(\overline{A}, Q) = e(\alpha a P, Q) = e(P, Q)^{\alpha a} = e(\alpha P, aQ) = e(\overline{P}, A_Q).$$

Similarly, equation (7.2) is satisfied by choice of the certificate (C_i, r_i)

$$\begin{aligned} e(\overline{K} + \overline{A} + \overline{B}, \overline{C}) &= e(\alpha(k_i P + A + r_i B), \beta C_i) = \\ &= e(k_i P + A + r_i B, C_i)^{\alpha\beta} = e(P, Q)^{\alpha\beta} = e(\overline{P}, \overline{Q}). \end{aligned}$$

By definition of the proof of knowledge the signature σ will verify. The final check is the revocation check. We know $\overline{K} = \alpha k_i P$, so $\overline{K} = k\overline{P}$ holds if and only if $k_i = k$. Hence, the response ξ is deemed valid if and only if $k \notin RL$. □

Lemma 7.8. *In the random oracle model there exists an extractor that can extract an SDH-pair by controlling a successful forger and the random oracle. If the forger is successful with probability ϵ and makes at most q_h hash queries and runs in time t then the extractor succeeds with probability $(\epsilon - 1/p)(\epsilon - 2/p)/(16q_h)$.*

Proof. The proof runs as follows. First we show how to obtain two signatures with different challenges. Then we show how we can extract the secrets and recover an SDH-pair.

Suppose we can rewind the forger in the self-blindable credential scheme. The forger sends $(\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma)$ with $\sigma = (e, s_\alpha, s_k, s_r, s_\beta)$ in response to a random nonce η sent by the verifier. The commitments are given by $R_\alpha = s_\alpha P - e\overline{P}, R_k = s_k P - e\overline{K}, R_r = s_r B - e\overline{B}, R_\beta = s_\beta Q - e\overline{Q}$. The challenge is given by $e = h(R_\alpha \parallel R_k \parallel R_r \parallel R_\beta \parallel \eta)$. The forking lemma of Pointcheval and Stern [68], see Theorem 3.33, shows that with probability at least $(\epsilon - 1/p)(\epsilon - 2/p)/(16q_h)$ the extractor can manipulate the forger and the random oracle to obtain a second response $(\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma')$ with $\sigma' = (e', s'_\alpha, s'_k, s'_r, s'_\beta)$, with $R_\alpha = s'_\alpha P - e'\overline{P}, R_k = s'_k P - e'\overline{K}, R_r = s'_r B - e'\overline{B}, R_\beta = s'_\beta Q - e'\overline{Q}$, so with the same commitments, but with a different challenge e'.

Let $\Delta e = e - e'$ and similarly for $\Delta s_\alpha, \Delta s_k, \Delta s_r, \Delta s_\beta$. Then we can recover the discrete logarithms by calculating:

$$\overline{\alpha} = -\frac{\Delta s_\alpha}{\Delta c}, \quad \overline{k} = -\frac{\Delta s_{\alpha k}}{\Delta c}, \quad \overline{r} = -\frac{\Delta s_{\alpha r}}{\Delta c}, \quad \overline{\beta} = -\frac{\Delta s_\beta}{\Delta c}. \tag{7.4}$$

The values $\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}$, sent by the forger, satisfy equations (7.1) and (7.2). From the definition of the non-interactive proof of knowledge σ and equation (7.4) it follows that $\overline{K} = \overline{k}P, \overline{B} = \overline{r}B, \overline{P} = \overline{\alpha}P$ and $\overline{Q} = \overline{\beta}Q$. Write $\overline{A} = \overline{a}P$, then from equation (7.1) it follows that $\overline{a} = \overline{\alpha}a$, hence $\overline{A} = \overline{a}P = \overline{\alpha}aP = \overline{\alpha}A_P$. Finally, let $\overline{C} = \overline{c}Q$, then from (7.2) it follows that $\overline{c}(\overline{k} + \overline{a} + \overline{r}b) = \overline{\alpha}\overline{\beta}$. By solving for $\overline{c}$ and dividing by $\overline{\beta}$ we find that:

$$\overline{c}/\overline{\beta} = \frac{1}{\overline{k}/\overline{\alpha} + a + \overline{r}/\overline{\alpha}b}.$$

Hence $(\overline{C}/\beta, \overline{k}/\overline{\alpha} + \overline{r}/\overline{\alpha}b)$ is an SDH-pair corresponding to the public key $A_P = aP$. Note that indeed all these values can be recovered from a successful forger. □

Theorem 7.9. *The self-blindable credential scheme in (G_1, G_2) has $(t, n, q_V, q_h, \epsilon)$ anonymity in the random oracle model assuming the (t, ϵ') Decision Diffie-Hellman assumption holds in the groups G_1 with $\epsilon' = \frac{\epsilon}{2}\left(\frac{1}{n^2} - \frac{q_V q_h}{p^4}\right)$ and the (t, ϵ) Decision Diffie-Hellman holds in G_2.*

Proof. Suppose algorithm $\mathcal{A}$ $(t, n, q_V, q_h, \epsilon)$-breaks the anonymity of the self-blindable credential scheme. We build an algorithm $\mathcal{B}$ that breaks the Decision Diffie-Hellman assumption in G_1. Algorithm $\mathcal{B}$ is given as input a 4 tuple $(P, X = xP, Y = yP, Z = zP) \in G_1^4$, where $P \in_R G_1$ and $x, y \in_R \mathbb{Z}_p$ and either z is random in $\mathbb{Z}_p$ or $z = xy$. Algorithm $\mathcal{B}$ decides whether $z = xy$ by interacting with $\mathcal{A}$ as follows.

Setup(n) Algorithm $\mathcal{B}$ sets P as generator for G_1 and chooses a random $Q \in G_2$ as generator for G_2. Now, acting as the certificate authority it generates private keys $a, b \in_R \mathbb{Z}_p$. Algorithm $\mathcal{B}$ sets $A_P = aP, A_Q = aQ, B = bP$ and sends P, Q, A_P, A_Q, B to algorithm $\mathcal{A}$.
All of the points are randomly generated, just as in the normal setup protocol. Hence, algorithm $\mathcal{A}$ cannot distinguish this from a normal run. Furthermore, $\mathcal{B}$ picks two random users $i_0, i_1 \in_R \{1, \ldots, n\}$ and keeps i_0, i_1 secret. For all users $i \neq i_0, i_1$ algorithm $\mathcal{B}$ picks a private key $k_i \in_R \mathbb{Z}_p$, generates a random value $r_i \in_R \mathbb{Z}_p$ and generates the corresponding certificate $C_i = 1/(k_i + a + r_i b)Q$. For users i_0 and i_1, it chooses a secret value $\gamma \in_R \mathbb{Z}_p$.

The intuition for the rest of the simulation is that distinguisher $\mathcal{B}$ will behave as if the private key of user i_0 is x, while the private key of i_1 is z/y. To do so, it sets $\overline{K}_{i_0} = X$, $\overline{P}_{i_0} = P$ and $\overline{K}_{i_1} = Z$, $\overline{P}_{i_1} = Y$, and multiplies them with a new blind $\alpha \in_R \mathbb{Z}_p$. Furthermore, $\mathcal{B}$ will behave as if $r_{i_0} = (\gamma - x - a)/b$ and $r_{i_1} = (\gamma - z/y - a)/b$. We emphasize that $\mathcal{B}$ does not know the value of these private keys nor can it directly calculate r_{i_0}, r_{i_1}. Observe that if $z = xy$ then the private keys are equal and $r_{i_0} = r_{i_1}$, but if $Z \in_R G_1$ then they are independent.

Hash queries At any time, $\mathcal{A}$ can query the hash function h for the hash of a value m. If m has not been queried before, $\mathcal{B}$ returns a new random value, otherwise $\mathcal{B}$ returns the same value as before.

Queries Algorithm $\mathcal{A}$ can issue prove queries and corruption queries. For queries for user $i \neq i_0, i_1$ algorithm $\mathcal{B}$ uses the private key k_i to generate a response as usual. For queries involving user i_0 or i_1 algorithm $\mathcal{B}$ responds as follows.

Prove Given a nonce η and a user $i \in \{i_0, i_1\}$ algorithm $\mathcal{B}$ must generate a valid response for η using user i's private key.

- To generate a response for user $i = i_0$, $\mathcal{B}$ picks blinds $\alpha, \beta \in_R \mathbb{Z}_p$, and starts making assignments:

$$\overline{K} = \alpha X, \quad \overline{P} = \alpha P, \quad \overline{A} = \alpha a P, \quad \overline{Q} = \beta Q.$$

  These values are indistinguishable from those produced by any other user. To produce a valid response $\mathcal{B}$ has to fake a certificate. To do so, it behaves as if $r_{i_0} = (\gamma - x - a)/b$, since then $\overline{B} = \alpha r_{i_0} B = \alpha(\gamma - x - a)P = \alpha(\gamma P - X - A)$. Note that r_{i_0} is uniformly distributed as a result of γ being uniformly distributed and hence indistinguishable from $\mathcal{A}$'s point of view. The remaining values are now given by:

$$\overline{B} = \alpha(\gamma P - X - A), \quad \overline{C} = \beta \frac{1}{\gamma} Q$$

- To generate a response for user $i = i_1$, $\mathcal{B}$ picks blinds $\alpha, \beta \in_R \mathbb{Z}_p$, and makes the following assignments:

$$\overline{K} = \alpha Z, \quad \overline{P} = \alpha Y, \quad \overline{A} = \alpha a Y, \quad \overline{Q} = \beta Q.$$

  From the perspective of $\mathcal{A}$ it is as if a blind $\alpha' = \alpha y$ was used, however, $\alpha \in_R \mathbb{Z}_p$ so $\mathcal{A}$ cannot use this to distinguish this tuple

from any other. To fake the certificate $\mathcal{B}$ behaves as if $r_{i_1} = (\gamma - z/y - a)/b$. Then $\overline{B} = \alpha y r_{i_1} B = \alpha(\gamma - z - ay)P = \alpha(\gamma Y - Z - aY)$. Also r_{i_1} is uniformly distributed due to γ. The remaining values are given by

$$\overline{B} = \alpha(\gamma Y - Z - aY) \quad \overline{C} = \beta \frac{1}{\gamma} Q$$

In both cases algorithm $\mathcal{B}$ has to provide the signature σ showing, in essence, the knowledge of both αr, αx_i and the blinds α and β. As described in section 3.7.1, algorithm $\mathcal{B}$ can fake the signature in the random oracle model as follows. First it chooses $s_{\alpha k}, s_{\alpha r}, s_\alpha, s_\beta, c \in_R \mathbb{Z}_p$ at random. This fixes the value $R_{\alpha k} = s_{\alpha k} P + cX$ and similar for $R_{\alpha r}, R_\alpha, R_\beta$. Finally, it sets $c = h(\eta \parallel R_{\alpha k} \parallel R_{\alpha r} \parallel R_\alpha \parallel R_\beta)$ by controlling the random oracle. In the unlikely event that $\mathcal{A}$ queried h on this value before, $\mathcal{B}$ aborts and reports failure. Since $s_{\alpha k}, s_{\alpha r}, s_\alpha, s_\beta, c$ are random, so are the R's, thus h was queried on this exact value with probability at most q_h/p^4.

Corrupt Algorithm $\mathcal{B}$ reports failure if $\mathcal{A}$ ever issues a corruption request for users i_0 or i_1.

Challenge Algorithm $\mathcal{A}$ outputs a nonce η and two users i_0^* and i_1^* where it wishes to be challenged. If $\{i_0^*, i_1^*\} \neq \{i_0, i_1\}$ then $\mathcal{B}$ reports failure and aborts. Otherwise, assume that $i_0^* = i_0$ and $i_1^* = i_1$. Algorithm $\mathcal{B}$ picks a random $d \in_R \{0, 1\}$ and generates a response for user i_d in the same way as for ordinary verification queries in the Query phase. It gives the response as the challenge to $\mathcal{A}$.

Restricted Queries Algorithm $\mathcal{A}$ issues restricted queries. Challenger $\mathcal{B}$ responds as in the Query phase.

Output Eventually, $\mathcal{A}$ outputs its guess $d' \in \{0, 1\}$ for d. If $d' = d$ then $\mathcal{B}$ indicates that Z is independent, otherwise it indicates that Z is dependent.

Suppose that $\mathcal{B}$ does not abort during the simulation. First consider the case where Z is random in G_1. We show that the responses produced for i_0 and i_1 are indistinguishable from independent responses. The responses restricted to points on G_1 are independent because they directly depend on the choice of $x, y, z \in \mathbb{Z}_p$. The values $(\overline{Q}, \overline{C})$ on G_2, however, do not depend on $x, y, z \in \mathbb{Z}_p$ and in fact the underlying certificate is equal for i_0 and i_1. If the adversary can detect this, then it can distinguish the tuple $((\alpha Q, \alpha c Q), (\hat{\alpha} Q, \hat{\alpha} c Q))$ from the tuple $((\alpha Q, \alpha c Q), (\hat{\alpha} Q, \hat{\alpha} c' Q))$. By the DDH assumption on G_2 this is not possible. So for Z random in G_1 the responses from i_0 and i_1 are indistinguishable from independent. So, distinguisher $\mathcal{B}$ simulates the anonymity game perfectly, and by assuption that the adversary breaks the anonimity of the game we have $Pr[d = d'] > \frac{1}{2} + \epsilon$.

When $z = xy$ the response that is sent as a challenge is independent of b; by choosing a different blind α we can transform a response for i_0 into one for i_1 and vice versa. It follows that $Pr[b = b'] = 1/2$ in this case. Therefore, assuming that $\mathcal{B}$ does not abort, it has advantage at least $\epsilon/2$ in solving the Decisional Diffie-Hellman problem.

Algorithm $\mathcal{B}$ aborts if the values i_0^* and i_1^* are not guessed correctly, furthermore, it can abort due to verification requests by $\mathcal{A}$. The probability that a

given verification request causes an abort is at most q_h/p^4 so the probability that that $\mathcal{B}$ aborts as a result of the verification queries is at most $q_V q_h/p^4$. If $\mathcal{B}$ does not abort during the query phase then $\mathcal{A}$ learns nothing about the true values of i_0, i_1, hence the probability that the choice of challenge do not cause $\mathcal{B}$ to abort is at least $1/n^2$. Since we require that $\mathcal{A}$ picks users i_0, i_1 we also know that $\mathcal{A}$ does not corrupt i_0 or i_1, hence $\mathcal{B}$ will not abort on any of the corruption queries. It follows that $\mathcal{B}$ solves the given Decisional Diffie-Hellman problem with advantage at least $\frac{\epsilon}{2}\left(\frac{1}{n^2} - \frac{q_V q_h}{p^4}\right)$. □

Theorem 7.10. *The self-blindable credential scheme in* (G_1, G_2) *is* $(t, n, q_V, q_h, \epsilon)$ *unforgeable in the random oracle model assuming the* (q, t', ϵ')*-Strong Diffie-Hellman assumption holds, where* $n = q, \epsilon = 4nq_h\sqrt{\epsilon'} + n/p$*, and* $t = \Theta(1)t'$.

Proof. Intuitively, the proof runs as follows. Given a forger $\mathcal{A}$ that breaks the unforgeability we construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the SDH-assumption. The algorithm $\mathcal{B}$ obtains an SDH-instance and creates a group of users with certificates based on this instance. Then it interacts with $\mathcal{A}$ as described in the unforgeability game. The forger $\mathcal{A}$, if successful, eventually outputs a valid response. Finally, algorithm $\mathcal{B}$ applies the extractor from theorem 7.8 to recover the SDH-pair from this response.

To obtain an SDH-pair for the SDH-instance the extracted SDH-pair needs to be new in the sense that algorithm $\mathcal{B}$ did not previously generate it. Two types of forgeries are possible: either the forged certificate is different from those algorithm $\mathcal{B}$ specified for the users, or it matches a certificate for one of the users. In the former case there is no problem; the certificate is always new. To, simultaneously, deal with the latter case algorithm $\mathcal{B}$ picks, at random, one user, i^*, for which it does not generate a certificate, but fakes one. It only tries to extract an SDH-pair if the forger produces a certificate it did not generate itself. We note that, whatever the type of forger, there is a non-negligible probability of obtaining a new certificate, either because it is a new certificate altogether, or because algorithm $\mathcal{B}$ guessed the user i^* correctly and it learns a certificate it did not know either.

In the following we describe how algorithm $\mathcal{B}$ creates the users from an SDH instance and how it interacts with forger $\mathcal{A}$.

Setup Algorithm $\mathcal{B}$ is given as input the bilinear group pair (G_1, G_2) and pairing e together with the n-SDH instance $(U, xU, x^2U, \ldots, x^nU, P, xP)$, with $P \in G_1$ and $U \in G_2$. The goal of $\mathcal{B}$ is to generate an SDH-pair $(1/(x+k)U, k)$, for a $k \in \mathbb{Z}_p$ of its choice. We follow Boneh and Boyen's Lemma 9 [13] to construct $n-1$ signatures from the SDH instance. Algorithm $\mathcal{B}$ generates $n-1$ distinct private keys $k_1, \ldots, k_{n-1} \in_R \mathbb{Z}_p$ and a secret key $b \in_R \mathbb{Z}_p^*$. For each k_i let $r_i \in_R \mathbb{Z}_p \setminus \{(k_i + x)/b\}$. Define the univariate polynomial f as $f(X) = \prod_{i=1}^{n-1}(X + k_i + r_i b)$. Expand f and write $f(X) = \sum_{i=0}^{n-1} f_i X^i$, with coefficients $f_i \in \mathbb{Z}_p$. Pick a random $\theta \in \mathbb{Z}_p^*$ and compute

$$Q = \theta f(x)U = \sum_{i=0}^{n-1} \theta f_i \left((x^i)U\right) \in G_2.$$

The points P and Q will be $\mathcal{B}$'s choice of generators for G_1 and G_2 respectively. Since θ is random these generators will be as well. Furthermore, $\mathcal{B}$ sets $A_P = xP$ and $B = bP$. The final system parameter, A_Q, is given by:

$$A_Q = xQ = \sum_{i=0}^{n-1} \theta f_i \left((x^{i+1}U\right) = \sum_{i=1}^{n} \theta f_{i-1} \left((x^i)U\right) \in G_2.$$

We would not have been able to calculate this value if we had tried to generate n signatures. We assume that $f(x) \neq 0$, if it is not we can recover x and create the desired SDH-tuple trivially.
For each $i = 1, \ldots, n-1$, we must create a signature $C_i = 1/(x+k_i+r_ib)Q$ over $k_i + r_ib$. Let $g_i(X) = f(X)/(X + k_i + r_ib) = \prod_{j=1, j\neq i}^{n-1}(X + k_j + r_jb)$, then $C_i = g_i(x)\theta U$. The value of the C_i can subsequently be determined by expanding $g_i(X)$ as before.
Algorithm $\mathcal{B}$ picks one user $i^* \in_R \{1, \ldots, n\}$ for which we will simulate the certificate. It generates random values $s_{i^*} \in_R \mathbb{Z}_p$ and $r_{i^*} \in_R \mathbb{Z}_p$ for this user. Next, algorithm $\mathcal{B}$ distributes the remaining $n-1$ signatures and the corresponding $n-1$ private keys over the other users by storing (k_i, C_i, r_i) accordingly. Finally, it sends the points $(P, Q, A_P, A_Q, B = bP)$ and a description of the groups to the forger $\mathcal{A}$.

Hash Queries At any time, $\mathcal{A}$ can query the hash function h for the hash of a value m. If m has not been queried before, $\mathcal{B}$ returns a new random value, otherwise $\mathcal{B}$ returns the same value as before.

The intuition for the following is that for all users except i^* algorithm $\mathcal{B}$ will behave normally. For user i^* algorithm $\mathcal{B}$ fakes a certificate by acting as if the public key is $K_{i^*} = s_{i^*}P - A_P$. We emphasize that $\mathcal{B}$ does not know the private key $s_{i^*} - x$ because it does not know x.

Queries Challenger $\mathcal{B}$ responds to the queries by $\mathcal{A}$ as follows.

Prove Algorithm $\mathcal{A}$ asks for a response by user i to a nonce η. If $i \neq i^*$ then $\mathcal{B}$ uses the private key k_i and certificate (C_i, r_i), to produce a valid response and returns this to $\mathcal{A}$.
Otherwise $i = i^*$. The challenger $\mathcal{B}$ fakes a response for the public key $K_{i^*} = s_{i^*}P - A_P$ by setting the values as follows:

$$\overline{K} = \alpha(s_{i^*}P - A_P), \quad \overline{P} = \alpha P, \quad \overline{A} = \alpha A, \quad \overline{B} = \alpha r_{i^*} B,$$
$$\overline{Q} = \beta Q, \quad \overline{C} = \frac{\beta}{s_{i^*} + br_{i^*}} Q.$$

Here we can fake a certificate because we have control over the public key B. The distinguisher controls the random oracle to fake the signature σ in the same way as in the anonymity proof. Note that the resulting response is indistinguishable from an unfaked response.

Corrupt Algorithm $\mathcal{A}$ asks for the private key and certificate of user i. If $i \neq i^*$, then $\mathcal{B}$ provides the requested values, otherwise it declares failure and exists.

Challenge Finally, algorithm $\mathcal{A}$ requests a nonce η from the challenger $\mathcal{B}$. Algorithm $\mathcal{B}$ generates $\eta \in_R \mathbb{Z}_p$ and sends it to $\mathcal{A}$.

Response If $\mathcal{A}$ is successful it outputs a valid response $\xi = (\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}, \sigma)$ to the nonce η.

Algorithm $\mathcal{B}$ now has to decide whether the certificate embedded in ξ is new. Let $\overline{K} = \overline{k}P, \overline{B} = \overline{r}B$ and $\overline{P} = \overline{\alpha}P$ then from theorem 7.8 we know that the extracted SDH-pair is given by $(\hat{\sigma}, \hat{k}) = (\overline{C}/\overline{\beta}, \overline{k}/\overline{\alpha} + \overline{r}/\overline{\alpha}b)$. For each of the pairs generated by $\mathcal{B}$ we know that $\hat{k}_i = k_i + r_i b$. Even though $\hat{k}$ is not known, we can use pairings to check whether $\hat{k} = \hat{k}_i$:

$$e(\overline{P}, (k_i + r_i b)Q) \stackrel{?}{=} e(\overline{K} + \overline{B}, Q).$$

Algorithm $\mathcal{B}$ reports failure if this equality holds for any user $i \neq i^*$, otherwise it returns ξ. If algorithm $\mathcal{B}$ returns, then automatically the signature in ξ is outside the forgers coalition. We follow the technique of Boneh and Boyen in [13] to transform this tuple into a solution to the posed n-SDH instance. We know that

$$\hat{\sigma} = \frac{1}{x + \hat{k}} Q = \frac{\theta f(x)}{x + \hat{k}} U.$$

We use long division to rewrite the polynomial f as $f(X) = (X + \hat{k})\gamma(X) + \gamma_*$ for some easily computable polynomial $\gamma(X) = \sum_{i=0}^{n-2} \gamma_i X^i$ and constant $\gamma_* \in \mathbb{Z}_p$. Here $\gamma_* \neq 0$, since $f(X) = \prod_{i=1}^{n-1}(X + k_i + r_i b)$ and $\gamma_* \neq k_i + r_i b$ for all i, thus $(X + \hat{k})$ does not divide $f(X)$. Use this expansion to rewrite the expression for $\hat{\sigma}$:

$$\hat{\sigma} = \theta \frac{f(x)}{x + \hat{k}} U = \theta \left(\frac{\gamma_*}{x + \hat{k}} + \sum_{i=0}^{n-2} \gamma_i x^i \right) U.$$

By dividing by θ and γ_* and correcting for the $\gamma(x)$ term we can calculate the following:

$$w = \frac{1}{\gamma_*} \left(\frac{\hat{\sigma}}{\theta} - \sum_{i=0}^{n-2} \gamma_i (x^i U) \right) = \frac{1}{\gamma_*} \left(\frac{\gamma_*}{x + \hat{k}} \right) U = \frac{1}{x + \hat{k}} U.$$

So we obtain the SDH-pair $(w, \hat{k})$ as solution to the original SDH-instance.

Algorithm $\mathcal{B}$ only returns a response if adversary $\mathcal{A}$ never corrupts user i^* and the signature in the response is new. As long as $\mathcal{A}$ never corrupts user i^* the simulation is perfect: responses for user i^* are indistinguishable from responses for other users. Let η be the probability with which the adversary produces a certificate that does not trace to any of the users. The probability of a new certificate is then $\eta + (1 - \eta)1/n$, so at least $1/n$. The probability that $\mathcal{A}$ never corrupts user i^* is at least $1/n$. So, with probability of at least ϵ/n^2 we obtain a response that contains a new certificate[2].

The succes probability of algorithm $\mathcal{B}$ is ϵ/n^2. From Theorem 7.8 it follows that the SDH-pair can be extracted with probability $\epsilon' = (\epsilon/n^2 - 1/p)^2/(16q_h^2)$ and in time $t' = \Theta(1)t$. □

[2]In the unlikely case that the adversary revokes all users this estimate is not valid. In this case we should just construct n valid signatures from an $(n+1)$-SDH problem. The adversary will then produce a certificate that does not trace to any user, and hence we recover the signature as normal. The success probability in this case is ϵ/n. Therefore ϵ/n^2 remains a valid lower bound.

Protocol	SBR	Emura	Chen & Li
Prover			
Nr. point mult. G_1	7	4	5
Nr. point mult. G_2	3	3	-
Nr. exponentiations G_T	-	-	4
Nr. modular mult.	6	1	5
Nr. points sent	10	5	4
Nr. of mod p	5	-	4
Verifier			
Nr. bilinear maps	4	$6 + 2\|RL\|$	1
Nr. point mult.	$8 + \|RL\|$	3	$4 + \|RL\|$
Properties			
Security proof	yes	no	yes
Revocation	yes	yes	yes
Pairing type	III	III	III

Table 7.1: Efficiency analysis of our new SBR protocol in comparison to the Emura protocol, see Section 6.2.2 and the Chen and Li group signature scheme, see Section 5.3.3.

7.5 Conclusions

In this chapter we have seen a new anonymous self-blindable credential system that supports revocation and has a security proof. Especially the revocation mechanism we use is a lot faster than the check used in the Emura protocol because only one point-multiplication is needed instead of two pairing calculations.

However, our protocol also shares some inherent limitations with the Emura and Chen and Li schemes, that originate from the Boneh and Boyen signature scheme. In order to generate the signature the issuer needs to know the private key. We do not know of a way to avoid this while still using Boneh and Boyen signatures.

Furthermore, due to the four zero-knowledge proofs, the amount of work done by the prover is more extensive. Table 7.1 shows an analysis of the the number of operations required. Why we choose this subdivision will become more apparent in the next chapter, but the main reason is that these operations have different durations. In the next chapter we shall also make some improvements to our protocol that will mitigate the large workload of the prover.

Chapter 8

Practical implementations

The protocol in the previous chapter is not as fast as the other self-blindable credential protocols we have seen before. As we saw, the zero-knowledge proofs make the protocol especially inefficient. In this chapter we will make our protocol more practical by adding two additional revocation options, by applying some heuristic transformations to our protocol to achieve a much faster protocol and by extending the show protocol so that it also works for multiple credentials at once.

8.1 How to revoke

The mechanism for revocation of a credential as described in the previous chapter was very straightforward: to revoke a user its private key is provided to all the verifiers as part of the revocation list. In practice this means that this key needs to be stored somewhere, for example at some trusted third party that will reveal this key on request. Another option would be to print the private key on the outside of the physical card. The user, or any that finds the card, can then revoke it by publishing this number. We note that knowing the private key does not immediately imply that anyone can use the card since the card itself might provide additional protection, but it is nevertheless a bit dangerous.

Considering the possible security issues we note that our protocol can easily be adapted to use an Emura et al.-like revocation scheme where instead of the private key k_i the value k_iQ is provided. The revocation check is then replaced by the pairing verification

$$e(\overline{K}, Q) \stackrel{?}{=} e(\overline{P}, k_iQ), \tag{8.1}$$

where k_iQ is an element from the revocation list. Unfortunately, this does mean that our fast point-multiplication check is replaced by two pairing calculations.

Both in the above modification, as well as in our original protocol, revocation is always on a per private key basis. However, suppose that an issuer grants multiple credentials over the same private key, then revocation on a per certificate basis would make more sense. In fact this is easily possible with our protocol. Instead of using k_iQ as a revocation we can use $c_iP = 1/(k_i + a + r_ib)P$ and use the following alternative to equation (8.1):

$$e(P, \overline{C}) \stackrel{?}{=} e(c_iP, \overline{Q}), \tag{8.2}$$

where c_iP is an element of the revocation list. We note that providing c_i by itself is not an option. Given $1/(k_i+a+r_ib)$ for a known private key k_i the user can recover $a + r_ib$ and hence make new certificates. This cannot be prevented either by also supplying r_i as a revocation token as we have no way of checking the value $\overline{B} = \alpha r_iB$ against it.

8.2 Protocol with proof of knowledge removed

Let us consider the zero-knowledge proof. In effect we only demonstrate that we know the values $\alpha, \alpha k_i, \alpha r_i, \beta$ corresponding to the points $\overline{P}, \overline{K}, \overline{B}, \overline{Q}$. This structure is reminiscent of a set of public-private key pairs. We shall now examine how we can replace this zero-knowledge proof by simple signatures involving the private keys. Recall that the original issue with this idea, that a verifier can now obtain a signature on any message of its choosing, is no longer relevant here, because the private keys are ephemeral.

We will use simple Chaum-Pedersen signatures as described in Section 3.6.1. Recall that here a message N is signed by multiplying it with the private key. As an example let k be the private key and $K = kP$ the corresponding public key. Then the signature by K on N is $\sigma = kN$. To make verification simple we consider the case where the verifier knows that the message N is of the form ηP for some nonce η chosen at random. Then it can verify the signature σ by checking that $\sigma = \eta K$. Notice that this also means that the verifier learns nothing by obtaining such a signature as it could easily have generated it by itself.

To replace the complete zero-knowledge proof the verifier sends three nonces $N_P = \eta_P P, N_B = \eta_B B, N_Q = \eta_Q Q$. The prover then creates the signatures $\overline{M}_k = \alpha k_i N_P, \overline{M}_r = \alpha r_i N_B, \overline{M}_\beta = \beta N_Q$. These signatures can be checked by verifying that $\overline{M}_k = \eta_P \overline{K}$, $\overline{M}_r = \eta_B \overline{B}$ and $\overline{M}_\beta = \eta_Q \overline{Q}$. An additional signature with α as private key is not needed as the prover already sends $\overline{A} = \alpha A$. The verifier can verify it use the pairing check $e(\overline{A}, Q) = e(\overline{P}, A_Q)$ instead. The simplified version of the protocol that is obtained in this way is shown in Figure 8.1.

The attentive reader might wonder why we can not simplify this to use a single nonce, i.e. $\eta_P = \eta_B = \eta_Q$. This simplification would allow the prover to cheat in the following way. It sets $\overline{K} = \alpha K - \gamma P$ and $\overline{B} = \alpha r_i B + \gamma P$ for some random value of $\gamma \in \mathbb{F}_p$. The pairing checks are still satisfied. Similarly, the prover can create valid signatures by setting $\overline{M}_k = (\alpha k - y)N_P$ and $\overline{M}_r = \alpha r_i N_B + y N_P$. This allows the prover to side-step the revocation check. The problem here is that by using the same nonce we are no longer guaranteed that the user actually knows the discrete logarithm of $\overline{B}$ with respect to B.

Prover		*Verifier*
		nonces $\eta_P, \eta_B, \eta_Q \in_R \mathbb{Z}_p$
into N_P, N_B, N_Q	$\longleftarrow$	**send** $\eta_P P, \eta_B B, \eta_Q Q$
blinding $\alpha, \beta \in_R \mathbb{Z}_p$		
send $\alpha K_i, \alpha A_P, \alpha r_i B, \alpha P, \beta Q, \beta C_i$	$\longrightarrow$	**into** $\overline{K}, \overline{A}, \overline{B}, \overline{P}, \overline{Q}, \overline{C}$
send $\alpha k_i N_P, \alpha r_i N_B, \beta N_Q$	$\longrightarrow$	**into** $\overline{M_k}, \overline{M_r}, \overline{M_\beta}$
		verify:
		$e(\overline{A}, Q) \stackrel{?}{=} e(\overline{P}, A_Q)$.
		$e(\overline{K} + \overline{A} + \overline{B}, \overline{C}) \stackrel{?}{=} e(\overline{P}, \overline{Q})$.
		$\overline{M_k} \stackrel{?}{=} \eta_P \overline{K}$.
		$\overline{M_r} \stackrel{?}{=} \eta_B \overline{B}$.
		$\overline{M_\beta} \stackrel{?}{=} \eta_Q \overline{Q}$.
		for every entry k
		in the revocation list:
		verify $k\overline{P} \stackrel{?}{\neq} \overline{K}$.

Figure 8.1: Practical self-blindable credential protocol with revocation. Here the proof of knowledge is replaced by three Chaum-Pedersen signatures. We will refer to this protocol as the SBRCP protocol.

We point out that we can now no longer extract the private keys from a prover, nor can we forge a signature by suitably rewinding the other party. Therefore, we can no longer proof the correctness of this transformed protocol. Nevertheless, the heuristic that being able to produce a signature on a random nonce is a good proof of identity [57].

8.3 Incorporating nonces

Comparing our protocol with Chaum-Pedersen signatures to the protocol proposed by Emura et al. we still note a conceptual difference in how the signature on the nonce is checked. In fact, in Emura et al.'s protocol the nonces are checked implicitly using the verification equations. In our protocol we can obtain a similar behavior, although we should emphasize that we do not know how to prove its security.

First, notice that the values $\overline{K}$ and $\overline{M_k}$ only differ by the factor induced by the nonce η_P. Similarly, $\overline{B}$ and $\overline{M_r}$ differ by a factor η_B, and $\overline{Q}$ and $\overline{M_\beta}$ differ by a factor η_Q. These factors are precisely used to verify the signatures. However, this can also be performed implicitly by using the values $\overline{M_k}, \overline{M_r}, \overline{M_\beta}$ in the verification equations. Instead of verifying that

$$e(\overline{K} + \overline{A} + \overline{B}, \overline{C}) = e(\overline{P}, \overline{Q})$$

we can immediately use the signatures by checking that

$$e(\overline{M_k}/\eta_P + \overline{A} + \overline{M_r}/\eta_B, \overline{C}) = e(\overline{P}, \overline{M_\beta}/\eta_Q).$$

Prover		*Verifier*
		nonces $\eta_P, \eta_B, \eta_Q \in_R \mathbb{Z}_p$
into N_P, N_B, N_Q	$\longleftarrow$	**send** $\eta_P P, \eta_B B, \eta_Q Q$
blinding $\alpha, \beta \in_R \mathbb{Z}_p$		
send $\alpha A_P, \alpha P, \beta C_i$	$\longrightarrow$	**into** $\overline{A}, \overline{P}, \overline{C}$
send $\alpha k_i N_P, \alpha r_i N_B, \beta N_Q$	$\longrightarrow$	**into** $\overline{M_k}, \overline{M_r}, \overline{M_\beta}$
		verify $e(\overline{A}, Q) \stackrel{?}{=} e(\overline{P}, A_Q)$.
		verify $e(\overline{M_k}/\eta_P + \overline{A} + \overline{M_r}/\eta_B, \overline{C})$ $\stackrel{?}{=} e(\overline{P}, \overline{M_\beta}/\eta_Q)$.
		for every entry k in the revocation list: verify $k\overline{P} \stackrel{?}{\neq} \overline{M_k}/\eta_P$.

Figure 8.2: Further optimization of our protocol where signatures $\overline{M_k}, \overline{M_r}, \overline{M_\beta}$ are directly used in the verification of the signature. This saves sending three points, as well as 3 point multiplications. We will refer to this protocol as the SBRCPShort protocol.

Note that the verifier cancels the nonces during verification. So, for a prover to successfully pass this test, it has to include nonces, and hence be able to create the signatures. We emphasize once more that this argument is primarily heuristic, but we do believe it is sound. The resulting protocol is shown in Figure 8.2

The efficiency of the two new protocols are analyzed in Table 8.1 and compared with the Emura protocol and the Chen and Li group signature scheme. We see that our new protocols are quite a bit more efficient then our original protocol. Furthermore, the last protocol seems to now be comparable to the group-signature scheme by Chen and Li.

8.4 Implementation on a smart card

In order to give a fair comparison of our new protocol with respect to smart card performance, we need to take into account the capabilities of a smart card. We did not implement our protocol on a smart card as we do not know of a card that supports point multiplications of points on a curve over an extension field, like G_2 in case of type III pairings. Furthermore, it turns out that often smart cards do not expose the modular multiplication unit, resulting in the counterintuitive result that a modular multiplication modulo a prime p can be much slower, due to an inefficient software implementation, than point-multiplication with a point of order P, that is implemented in hardware. In this section we examine some sources that allow us to estimate duration of our showing protocol and hence compare it against other protocols.

Our first source is Hoepman et al. [45] who give a description of how their protocol can be implemented on a Java Card smart card. The Java Card API

Protocol	SBR Figure 7.1	SBRCP Figure 8.1	SBRCPShort Figure 8.2	Emura Figure 6.4	Chen & Li Section 5.3.3
Prover					
Nr. point mult. G_1	7	6 (8)	4 (6)	4	5
Nr. point mult. G_2	3	3	2	2	-
Nr. exponentiations G_T	-	-	-	-	4
Nr. modular mult.	6	2 (0)	2 (0)	0	5
Nr. random values	6	2	2	1	5
Nr. points sent G_1	7	6	4	3	4
Nr. points sent G_2	3	3	2	2	-
Nr. of mod p	4	0	0	0	5
Verifier					
Nr. bilinear maps	4	4	4	$6 + 2\|RL\|$	1
Nr. point mult.	$8 + \|RL\|$	$3 + \|RL\|$	$3 + \|RL\|$	3	$4 + \|RL\|$
Properties					
Security proof	yes	no	no	no	yes
Revocation	yes	yes	yes	yes	yes
Pairing type	III	III	III	III	III

Table 8.1: Efficiency analysis of our new protocols in comparison to the Emura protocol and the Chen and Li group signature. The SBR protocol is our protocol with zero-knowledge proofs as demonstrated in the previous chapter, this protocol leads to the SBRCP protocol by replacing the signatures using Chaum-Pedersen signatures while finally the SBRCPShort version is obtained by directly incorporating these signatures into the verification. The numbers in parentheses suggest an alternative counting where we optimize for smart cards.

	Time (ms)	Source
Blind generation	379	[45]
Point multiplication G_1	98	[45]
Point multiplication G_2	1000 - 3000	Estimated based on [63]
Point multiplication G_T	430	Assumption based on [11]
Modulo p multiplication	$\approx$ 1700 (10)	Estimated from [5]

Table 8.2: Time estimates for common operations on smart card. In all these operations the group order is 192 bits. A side effect of generating a blind α is a point αZ for a given generator Z. The time in parentheses for a modulo p multiplication is a rough estimate for the running time if it would be implemented in hardware instead of in software.

only exposes the cryptographic co-processor via a number of high-level functions. Hoepman et al. describe how the Diffie-Hellman key agreement function and the Diffie-Hellman key generation functions can be used to perform some operations in our protocol. Both are used in the Diffie-Hellman key-exchange protocol in Section 3.4. The Diffie-Hellman key generation function is used to generate an ephemeral key k_A and a corresponding public key $k_A P$. The generator P can be chosen by the user. Thus this function is useful for generating a blind as well as the blinded point. The Diffie-Hellman key agreement function generates the resulting public key $k_A k_B P$ in the Diffie-Hellman key-exchange protocol based on $k_A, B = k_B P$, and hence performs a point multiplication. Table 8.2 shows the required running time to complete these operations on a Java Card.

In 2006 Page et al. [63] compared some type III curves, with embedding degree 6, to type I curves in terms of computational complexity. Analyzing the results shows that a point multiplication in G_2 for a type III curve is about 10 to 30 times more expensive than a similar point multiplication in G_1. This ratio is based on a software implementation on a modern desktop CPU. In the following we take the slightly optimistic time of 2000 ms as an estimate, so we use a factor 20, for the time it takes a smart card to compute this point multiplication.

Finally, we need to estimate the time complexity of exponentiations in G_T, i.e. the group of p-th roots of unity. As far as we know this operation is not supported on the smart cards either. The p-th roots of unity lie in the field $\mathbb{F}_{p^6}$ for our curve, hence a software implementation would take at least 10 seconds per exponentiation in this field. Since we also plead for a hardware based implementation for point multiplication in G_T we assume a length of 430 ms for exponentiation in G_T, since that is the time it takes to do an exponentiation of a number modulo a 1280 bits prime.

Based on the estimates in Table 8.2 we can give a lower bound for the time it takes a smart card to run the verification protocol. Since this does not include any actual testing for our protocol nor for the Chen and Li group signatures we cannot accurately estimate the processing overhead. The results are shown in Table 8.3. We included a couple of other credential systems for which time estimates on a smart card are also given.

	Time (s)	Estimated	Revocation	Proof
Batina et al. [5, 45]	0.8	no [45]	no	no
Chen and Li [29]	13.9 (3.5)	yes	yes	yes
Idemix [21]	7.4	no [11]	no	yes
SBR protocol	17.5 (7.4)	yes	yes	yes
SBRCP protocol	7.5	yes	yes	no
SBRCPShort protocol	5.3	yes	yes	no

Table 8.3: Running times (or estimates thereof) for running the verification algorithm on a standard Java Smart Card. None of these numbers include the running time of the verifier. The time estimates in parenthesis are with the hardware modulo p multiplication. Further indicated are whether this values is estimated, if the method supports revocation and whether there exist a proof of security for the method.

8.5 Multiple credentials

Our protocol with implicit Chaum-Pedersen certificates can easily and efficiently be augmented to show multiple credentials simultaneously to a verifier. This speedup is achieved by using the same blinds for all the credentials. Suppose a user has a credential (C_1, r_1) for a private key k_1 and a credential (C_2, r_2) for a private key k_2. It can then show possession of these in a single protocol run as shown in Figure 8.3. Clearly, this scheme can be extended to demonstrate possession of additional credentials. For each new credential the prover needs to do 2 additional point-multiplications in G_1 and 1 additional point-multiplication in G_2. The verifier needs to do two additional pairing calculations.

If $k_1 = k_2$ then the combined protocol becomes even more simple, and allows us to prove possession of two certificates belonging to the same private key. This multi-show protocol does allow us to give an efficient implementation of the VerifyCredOnNym procedure from our ideal model. Thus giving rise to a complete credential system.

8.6 Backward unlinkability

Having backward unlinkability would be a nice addition to our protocol. Unfortunately, we have not been able to make this work. The solution we sketch here, which is the best we came up with, can be easily reduced the the ordinary protocol. Hence, it does not offer backward unlinkability. Nevertheless we display it here to provide a starting point for future work on this topic.

The main idea is to introduce epochs like Nakanishi and Funabiki [62]. Each epoch all of the entries on the revocation list are updated. Intuitively, making these entries dependent on the current epoch, ensures that the verifier cannot examine earlier transcripts with respect to a current revocation token, unless the verifier can somehow take out the epoch dependency. We will shortly show that it can, but first we give a sketch of the method. This sketch is an extension of our original method.

Prover		*Verifier*
		nonces $\eta_P, \eta_B, \eta_Q \in_R \mathbb{Z}_p$
into N_P, N_B, N_Q	$\longleftarrow$	**send** $\eta_P P, \eta_B B, \eta_Q Q$
blinding $\alpha, \beta \in_R \mathbb{Z}_p$		
send $\alpha A_P, \alpha P, \beta N_Q$	$\longrightarrow$	**into** $\overline{A}, \overline{P}, \overline{M_\beta}$
send $\alpha k_1 N_P, \alpha r_1 N_B, \beta C_1$	$\longrightarrow$	**into** $\overline{M_{k_1}}, \overline{M_{r_1}}, \overline{C_1}$
send $\alpha k_2 N_P, \alpha r_2 N_B, \beta C_2$	$\longrightarrow$	**into** $\overline{M_{k_2}}, \overline{M_{r_2}}, \overline{C_2}$
		verify $e(\overline{A}, Q) \stackrel{?}{=} e(\overline{P}, A_Q)$.
		verify $e(\overline{M_{k_1}}/\eta_P + \overline{A} + \overline{M_{r_1}}/\eta_B, \overline{C_1})$ $\stackrel{?}{=} e(\overline{P}, \overline{M_\beta}/\eta_Q)$.
		verify: $e(\overline{M_{k_2}}/\eta_P + \overline{A} + \overline{M_{r_2}}/\eta_B, \overline{C_2})$ $\stackrel{?}{=} e(\overline{P}, \overline{M_\beta}/\eta_Q)$.
		for every entry k in the revocation list: verify $k\overline{P} \stackrel{?}{\neq} \overline{K}/\eta_P$.

Figure 8.3: Showing two credentials (C_1, r_1) and (C_2, r_2) corresponding to the respective private keys k_1 and k_2 to a verifier simultaneously. A speedup is obtained by using the same random nonces.

For each epoch t a fresh random $\epsilon_t \in \mathbb{Z}_p$ is generated. This values is kept secret. With every change of epoch, a new revocation list is generated. First, $P_t = \epsilon_t P$ is associated with the revocation list. Then, for each revoked private key k an entry $Q_k = k\epsilon_i Q$ is added to the revocation list RL_i. Note that ϵ_i is not publicly known, while the revocation list RL itself is. We shall shortly see that an adversary can recover $\epsilon_i Q$, but for now consider this difficult. Therefore, the verifier gets a random point $X \in G_2$ and the point $X_t = \epsilon_i X$.

The protocol is extended in the following manner. The verifier sends the additional challenge P_t to the prover, who responds with the value $\overline{T} = \alpha k_i P_t$ (which should equal $\epsilon_t \overline{K}$). The verifier checks this with the test $e(\overline{T}, X) = e(\overline{K}, X_t)$. Finally, the verifier checks that the key is not revoked by checking, for every entry $Q_k \in RL$ that

$$e(\overline{P}, Q_k) \stackrel{?}{\neq} e(\overline{T}, Q),$$

which is basically an augmented version of the alternative revocation check in equation (8.1).

To see where the problem lies we need to first augment the security games. First we extend the games to have multiple epochs, this is rather easy. However, for the backward-unlinkability game we run into trouble. To model that the system is backward unlinkable we have to allow the following situation. The adversary requests revocation tokens of users 1 and 2 (this choice is without loss of generality) at epoch t, i.e. it obtains $Q_{k_1} = k_1\epsilon_t P$ and $Q_{k_2} = k_2\epsilon_t P$. It then asks to be challenged on these two users, but on a time $t' < t$. Clearly, if the

system is backward unlinkable then the adversary cannot win this. However, it can. First, it requests the revocation token of user 3 and gets $Q_{k_3} = k_3\epsilon_t Q$. Then, it corrupts user 3 and hence obtains k_3. Therefore, it can now calculate $Q_{k_3}/k_3 = \epsilon_t Q$. As a challenge it receives a values $\overline{K} = \alpha k_* P$ and $\overline{P} = \alpha P$, for some k_*. It then performs the following check:

$$e(\overline{P}, Q_{k_1}) \stackrel{?}{=} e(\overline{K}, \epsilon_t Q),$$

which clearly holds if and only if $k_* = k_1$, so this protocol is not backward unlinkable.

We note that even without this problem we significantly need to update our proofs. While it is clearly possible to provide the blinded values during the proofs if we can choose ϵ_i, this is not possible for the revocation tokens. These we cannot provide given only the DDH instance, while we should to complete the proof. Furthermore, the requirement that we can corrupt a revoked user is not unreasonable, and in fact may be a milder requirement than corrupting unrevoked users.

8.7 Conclusions

In this chapter we have seen a number of practical improvements to our protocol. These resulted in the final SBRCPShort protocol that is significantly faster than our original protocol, at the cost of no longer being provably correct, even though there is a clear heuristic leading to it. The efficiency analysis suffers from a lack of an actual implementation, but does show that when elliptic curve arithmetic over extension fields are implemented on smart cards our protocols form a viable alternative to existing credential systems and has comparable performance.

Chapter 9

Conclusions

In this chapter we will summarize our major results from this thesis. In addition to this we look ahead to possible future work that can be used to improve self-blindable credentials.

9.1 Conclusions

We set out to find a way to add revocation to the self-blindable credentials by Verheul. To this end we examined many techniques for adding revocation to a credential system; each with their own drawbacks and advantages. In some the computational load or communication cost of the user is very high, while in others this load is shifted solely to the verifier. Finally, in our new credential system we used a revocation mechanism that moves most of the load to the verifier, but nevertheless only needs one point-multiplication per item on the revocation list. This is considerably faster than the pairing revocation checks that were used in earlier attempts at revocable self-blindable credentials.

In order to accomplish our secondary goal we first derived a security model for our credential system. We gave precise definitions for the unforgeability, anonymity and revocability of a self-blindable credential scheme. Based on this we constructed a revocable self-blindable credential protocol that uses the afore mentioned fast revocation check. Contrary to earlier self-blindable credential schemes, this scheme is provably secure system in the random oracle model. As a side-effect of this effort we discovered some serious flaws in these earlier attempts, effectively making these protocols non-anonymous.

In our quest for a faster protocol we showed in chapter 8 how our formal protocol can be transformed to obtain a much faster protocol, at the cost of losing the proof of security.

We saw that a general disadvantage of using Boneh and Boyen signature in self-blindable credential systems is that some points need to lie on an elliptic curve over an extension field. Currently, there is no smart card support for

these curves, thus no practical implementation is possible at the time of writing. Nevertheless, we believe that the idea of self-blindable credentials forms a viable and simpler alternative to other, zero-knowledge based, credential systems.

9.2 Further work

A first challenge in our credential system is the issuing protocol. Currently, the issuer that issues a credential will always know the corresponding private key. And, by knowing this key, can trace this credential as well as revoke it. In practical situations more flexibility might be warranted to reduce the trust assumption on the issuer. Two possibilities would be a system in which (1) the user can choose their private key in secret, while the issuer still receives the revocation token; and (2) the issuer and user produce a revocation token that can only be used by a third party. It is interesting to see to what extend these properties can be obtained by creating a different self-blindable credential with possibly another underlying certificate scheme.

Another issue with our protocol is that it does not support backward unlinkability. After a credential or user has been revoked all its past transactions become traceable as well. We proposed some preliminary work in the direction of backward unlinkability, but this quickly falls apart if we allow for cheating users and verifiers. Nevertheless, we feel that backward unlinkability is an important property to have in a credential system, therefore more research in this direction would be warranted.

Furthermore, to use our protocol with smart cards we need support for point-multiplication on elliptic curves over extension fields. Further (literature) study could yield better estimates for the complexity of implementing this in hardware as well as for software implementation. In addition, it might be interesting to study other, faster, 16 or 32 bits smart cards, and investigate how much improvement can be obtained by using them. It may be possible be that the recent results by Mostowski and Vullers [60] already allow for faster implementations of our self-blindable protocols.

From a theoretical point of view it is interesting to determine whether the construction in Chapter 8 where Chaum-Pedersen signatures replace Schnorr proofs of identity can be provided with a solid foundation. Related to this problem is the question to find out what exactly are the differences between a zero-knowledge proofs of identity and the constructions, both with and without revocation, that we have studied in this thesis.

Bibliography

[1] Gergely Alpár, Jaap-Henk Hoepman, and Johanneke Siljee. *The Identity Crisis. Security, Privacy and USability Issues in Idenity Management.* Jan. 2011 (cit. on p. 53).

[2] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. "A Practical and Provably Secure Coalition-Resistant Group Signature Scheme". In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings.* Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Springer, 2000, pp. 255–270. ISBN: 3-540-67907-3 (cit. on p. 71).

[3] Giuseppe Ateniese, Dawn Song, and Gene Tsudik. "Quasi-efficient revocation of group signatures". In: *Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda, March 11-14, 2002, Revised Papers.* Ed. by Matt Blaze. Vol. 2357. Lecture Notes in Computer Science. Springer, 2003, pp. 183–197. ISBN: 3-540-00646-X (cit. on pp. 71, 72, 75).

[4] Niko Baric and Birgit Pfitzmann. "Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees". In: *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding.* Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 480–494. ISBN: 3-540-62975-0 (cit. on pp. 75–77).

[5] Lejla Batina et al. "Developing Efficient Blinded Attribute Certificates on Smart Cards via Pairings". In: *Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, Passau, Germany, April 14-16, 2010. Proceedings.* Ed. by Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny. Vol. 6035. Lecture Notes in Computer Science. Springer, 2010, pp. 209–222. ISBN: 978-3-642-12509-6 (cit. on pp. 9, 44, 70, 85, 110, 111).

[6] F.L. Bauer. *Decrypted secrets: methods and maxims of cryptology.* 4th. Springer, 2006. ISBN: 978-3540245025 (cit. on p. 14).

[7] Mihir Bellare and Oded Goldreich. "On Defining Proofs of Knowledge". In: *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*. Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1993, pp. 390–420. ISBN: 3-540-57340-2 (cit. on pp. 29, 30).

[8] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *Proceedings of the 1st ACM Conference on Computer and Communications Security - CCS '93, November 3-5, 1993, Fairfax, Virginia, USA*. ACM, 1993, pp. 62–73 (cit. on pp. 47, 48).

[9] Mihir Bellare, Haixia Shi, and Chong Zhang. "Foundations of Group Signatures: The Case of Dynamic Groups". In: *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*. Ed. by Alfred Menezes. Vol. 3376. Lecture Notes in Computer Science. Springer, 2005, pp. 136–153. ISBN: 3-540-24399-2 (cit. on pp. 70, 71, 74).

[10] Josh Benaloh and Michael de Mare. "One-Way Accumulators: A Decentralized Alternative to Digital Signatures". In: *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer, 1994, pp. 274–285. ISBN: 3-540-57600-2 (cit. on pp. 75, 76).

[11] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup. "Anonymous credentials on a standard java card". In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. Ed. by Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis. ACM, 2009, pp. 600–610. ISBN: 978-1-60558-894-0 (cit. on pp. 9, 70, 110, 111).

[12] Dan Boneh and Xavier Boyen. "Short Signatures without Random Oracles". In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 56–73. ISBN: 3-540-21935-8 (cit. on pp. 47, 73).

[13] Dan Boneh and Xavier Boyen. "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups". In: *J. Cryptol.* 21 (2 Feb. 2008), pp. 149–177. ISSN: 0933-2790 (cit. on pp. 45, 101, 103).

[14] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*. Ed. by Matthew K. Franklin. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 41–55. ISBN: 3-540-22668-0 (cit. on pp. 72, 73).

[15] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing". In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 213–229. ISBN: 3-540-42456-3 (cit. on p. 44).

[16] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing". In: *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*. Ed. by Colin Boyd. Vol. 2248. Lecture Notes in Computer Science. Springer, 2001, pp. 514–532. ISBN: 3-540-42987-5 (cit. on pp. 44, 45).

[17] Dan Boneh and Hovav Shacham. "Group signatures with verifier-local revocation". In: *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*. Ed. by Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel. ACM, 2004, pp. 168–177. ISBN: 1-58113-961-6 (cit. on pp. 73–75, 97).

[18] S. Brands. *A technical overview of digital credentials*. At http://www. credentica. com/technology/technology. html. 2002 (cit. on pp. 8, 67).

[19] Ernest F. Brickell, Daniel M. Gordon, Kevin S. McCurley, and David B. Wilson. "Fast exponentiation with precomputation". In: *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*. Ed. by Rainer A. Rueppel. Vol. 658. Lecture Notes in Computer Science. Springer, 1993, pp. 200–207. ISBN: 3-540-56413-6 (cit. on p. 74).

[20] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. "An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials". In: *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*. Ed. by Stanislaw Jarecki and Gene Tsudik. Vol. 5443. Lecture Notes in Computer Science. Springer, 2009, pp. 481–500. ISBN: 978-3-642-00467-4 (cit. on pp. 75, 76, 78).

[21] Jan Camenisch and Anna Lysyanskaya. "An efficient system for non-transferable anonymous credentials with optional anonymity revocation". In: *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*. Ed. by Birgit Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Springer, 2001, pp. 93–118. ISBN: 3-540-42070-3 (cit. on pp. 8, 53, 66, 111).

[22] Jan Camenisch and Anna Lysyanskaya. "Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials". In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer

Science. Springer, 2002, pp. 61–76. ISBN: 3-540-44050-X (cit. on pp. 72, 75–78).

[23] Jan Camenisch and Anna Lysyanskaya. "Signature Schemes and Anonymous Credentials from Bilinear Maps". In: *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*. Ed. by Matthew K. Franklin. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 1–6. ISBN: 3-540-22668-0 (cit. on p. 73).

[24] Jan Camenisch and Markus Stadler. "Efficient group signature schemes for large groups". In: *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. Ed. by Burton S. Kaliski Jr. Vol. 1294. Lecture Notes in Computer Science. Springer, 1997, pp. 410–424. ISBN: 3-540-63384-7 (cit. on p. 35).

[25] Jan Camenisch and Gregory M. Zaverucha. "Private Intersection of Certified Sets". In: *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*. Ed. by Roger Dingledine and Philippe Golle. Vol. 5628. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 108–127 (cit. on pp. 79–81).

[26] David Chaum. "Security Without Identification: Transaction Systems to Make Big Brother Obsolete". In: *Commun. ACM* 28.10 (1985), pp. 1030–1044 (cit. on pp. 53, 57, 83).

[27] David Chaum and Eugne van Heyst. "Group Signatures". In: *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*. Ed. by Donald W. Davies. Vol. 547. Lecture Notes in Computer Science. Springer, 1991, pp. 257–265. ISBN: 3-540-54620-0 (cit. on p. 70).

[28] David Chaum and Torben P. Pedersen. "Wallet Databases with Observers". In: *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*. Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1992, pp. 89–105. ISBN: 3-540-57340-2 (cit. on p. 44).

[29] Liqun Chen and Jiangtao Li. "VLR Group Signatures with Indisputable Exculpability and Efficient Revocation". In: *Proceedings of the 2010 IEEE Second International Conference on Social Computing, SocialCom / IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT 2010, Minneapolis, Minnesota, USA, August 20-22, 2010*. Ed. by Ahmed K. Elmagarmid and Divyakant Agrawal. IEEE Computer Society, 2010, pp. 727–734. ISBN: 978-0-7695-4211-9 (cit. on pp. 71, 73–75, 92, 111).

[30] Ivan Damgrard. *On Σ-protocols*. Lecture notes for CPT 2010, v.2. 2010 (cit. on p. 34).

[31] Ivan Damgrard and Jesper Buus Nielsen. *Commitment-schemes and Zero-knowledge Protocols*. An introduction. 2007. URL: `http://www.daimi.au.dk/~ivan/ComZK07.pdf` (cit. on pp. 35, 36).

[32] Luuk Danes. "Smart card integration in the pseudonym system Idemix". MA thesis. University of Groningen, 2007 (cit. on pp. 69, 70).

[33] W. Diffie and M. Hellman. "New directions in cryptography". In: *Information Theory, IEEE Transactions on* 22.6 (1976), pp. 644–654. ISSN: 0018-9448 (cit. on p. 40).

[34] David S. Dummit and Richard M. Foote. *Abstract Algebra*. 3rd. John Wiley & Sons, 2003 (cit. on p. 28).

[35] T. Elgamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *Information Theory, IEEE Transactions on* 31.4 (July 1985), pp. 469 –472 (cit. on pp. 18, 41, 80).

[36] Keita Emura, Atsuko Miyaji, and Kazumasa Omote. "A Certificate Revocable Anonymous Authentication Scheme with Designated Verifier". In: (2009), pp. 769–773 (cit. on pp. 44, 47, 86, 87, 89, 91).

[37] Amos Fiat and Adi Shamir. "How to prove yourself: practical solutions to identification and signature problems". In: *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1987, pp. 186–194. ISBN: 0-387-18047-8 (cit. on pp. 37, 47).

[38] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient Private Matching and Set Intersection". In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 1–19. ISBN: 3-540-21935-8 (cit. on pp. 79, 80).

[39] Jun Furukawa and Hideki Imai. "An Efficient Group Signature Scheme from Bilinear Maps". In: *Information Security and Privacy, 10th Australasian Conference, ACISP 2005, Brisbane, Australia, July 4-6, 2005, Proceedings*. Ed. by Colin Boyd and Juan Manuel González Nieto. Vol. 3574. Lecture Notes in Computer Science. Springer, 2005, pp. 455–467. ISBN: 3-540-26547-3 (cit. on pp. 73, 74).

[40] S. D. Galbraith. "Pairings". In: *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005. Chap. IX (cit. on p. 41).

[41] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for cryptographers". In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3113–3121 (cit. on pp. 42, 45).

[42] O. Goldreich. *Foundations of cryptography: basic tools*. Foundations of Cryptography. Cambridge University Press, 2001. ISBN: 9780521791724 (cit. on pp. 28–30, 32, 33).

[43] Oded Goldreich. *A Short Tutorial of Zero-Knowledge*. 2010 (cit. on p. 31).

[44] G.H. Hardy, E.M. Wright, D.R. Heath-Brown, and J.H. Silverman. *An introduction to the theory of numbers*. Oxford mathematics. Oxford University Press, 2008. ISBN: 9780199219865 (cit. on p. 15).

[45] Jaap-Henk Hoepman, Bart Jacobs, and Pim Vullers. "Privacy and Security Issues in e-Ticketing – Optimisation of Smart Card-based Attribute-proving". In: *Workshop on Foundations of Security and Privacy, FCS-PrivMod 2010, Edinburgh, UK, July 14-15, 2010. Proceedings*. Ed. by Veronique Cortier, Mark Ryan, and Vitaly Shmatikov. July 2010 (cit. on pp. 85, 86, 108, 110, 111).

[46] Susan Hohenberger and Stephen Weis. "Honest-Verifier Private Disjointness Testing Without Random Oracles". In: *Privacy Enhancing Technologies*. Ed. by George Danezis and Philippe Golle. Vol. 4258. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, pp. 277–294 (cit. on pp. 79, 80).

[47] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 2nd. Addison Wesley, 2000 (cit. on p. 29).

[48] R. Housley, W. Polk, W. Ford, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC 3280)*. United States, 2002 (cit. on p. 65).

[49] ITU-T. *Information technology Open systems interconnection The Directory: Public-key and attribute certificate frameworks*. Nov. 2008 (cit. on p. 63).

[50] Antoine Joux. "A One Round Protocol for Tripartite Diffie-Hellman". In: *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*. Ed. by Wieb Bosma. Vol. 1838. Lecture Notes in Computer Science. Springer, 2000, pp. 385–394. ISBN: 3-540-67695-3 (cit. on pp. 41, 43).

[51] Antoine Joux. "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems". In: *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings*. Ed. by Claus Fieker and David R. Kohel. Vol. 2369. Lecture Notes in Computer Science. Springer, 2002, pp. 20–32. ISBN: 3-540-43863-7 (cit. on p. 41).

[52] S. Kiyomoto and T. Tanaka. "Anonymous attribute authentication scheme using self-blindable certificates". In: *Intelligence and Security Informatics, IEEE ISI 2008 International Workshops: PAISI, PACCF, and SOCO 2008, Taipei, Taiwan, June 17, 2008. Proceedings*. Ed. by Christopher C. Yang et al. Vol. 5075. Lecture Notes in Computer Science. Springer, 2008, pp. 215 –217. ISBN: 978-3-540-69136-5 (cit. on pp. 44, 86, 87).

[53] N. Koblitz and A.J. Menezes. "A survey of public-key cryptosystems". In: *SIAM review* 46.4 (2004), pp. 599–634. ISSN: 0036-1445 (cit. on p. 40).

[54] Neal Koblitz. "Elliptic Curve Cryptosystems". In: *Mathematics of Computation* 48.177 (Jan. 1987), pp. 203–209 (cit. on p. 39).

[55] Jiangtao Li, Ninghui Li, and Rui Xue. "Universal Accumulators with Efficient Nonmembership Proofs". In: *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*. Ed. by Jonathan Katz and Moti Yung. Vol. 4521. Lecture Notes in Computer Science. Springer, 2007, pp. 253–269. ISBN: 978-3-540-72737-8 (cit. on p. 78).

[56] A. Menezes. "An introduction to pairing-based cryptography". In: *Recent trends in cryptography: UIMP-RSME Santaló Summer School, July 11-15, 2005, Universidad Internacional Menéndez Pelayo, Santander, Spain*. Ed. by Ignacio Luengo. Vol. 30. American Mathematical Society, 2009. ISBN: 0821839845 (cit. on p. 41).

[57] A.J. Menezes, T. Okamoto, and S.A. Vanstone. "Reducing elliptic curve logarithms to logarithms in a finite field". In: *Information Theory, IEEE Transactions on* 39.5 (1993), pp. 1639–1646. ISSN: 0018-9448 (cit. on pp. 14, 40, 107).

[58] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996 (cit. on pp. 14, 15, 24).

[59] Victor S Miller. "Use of elliptic curves in cryptography". In: *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*. Ed. by Hugh C. Williams. Vol. 218. Lecture Notes in Computer Science. Springer, 1986, pp. 417–426. ISBN: 0-387-16463-4 (cit. on p. 39).

[60] Wojciech Mostowski and Pim Vullers. "Efficient U-Prove Implementation for Anonymous Credentials on Smart Cards". In: *7th International ICST Conference on Security and Privacy in Communication Networks, SecureComm 2011, London, UK, September 7-9, 2011. Proceedings*. Ed. by George Kesidis and Haining Wang. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Tele-communications Engineering (LNICST). (to appear). Springer-Verlag, Sept. 2011 (cit. on pp. 70, 116).

[61] M. Myers, R. Anknew, S. Galperin, and C. Adams. *Internet public key infrastructure online certificate status protocol - OCSP*. Internet Request for Comments: RFC 2560. 1999 (cit. on p. 65).

[62] Toru Nakanishi and Nobuo Funabiki. "Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps". In: *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*. Ed. by Bimal K. Roy. Vol. 3788. Lecture Notes in Computer Science. Springer, 2005, pp. 533–548. ISBN: 3-540-30684-6 (cit. on pp. 73, 74, 111).

[63] D. Page, N.P. Smart, and F. Vercauteren. "A comparison of MNT curves and supersingular curves". In: *Applicable Algebra in Engineering, Communication and Computing* 17.5 (2006), pp. 379–392 (cit. on p. 110).

[64] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 223–238. ISBN: 3-540-65889-0 (cit. on p. 80).

[65] K. G. Paterson. "Cryptography from Pairings". In: *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005. Chap. X (cit. on p. 41).

[66] Andreas Pfitzmann and Marit Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. v0.34. Aug. 2010. URL: `http://dud.inf.tu-dresden.de/literatur/Anon\_Terminology\_v0.34.pdf` (cit. on pp. 56, 57).

[67] S. Pohlig and M. Hellman. "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.)" In: *Information Theory, IEEE Transactions on* 24.1 (1978), pp. 106–110. ISSN: 0018-9448 (cit. on p. 40).

[68] David Pointcheval and Jacques Stern. "Security Arguments for Digital Signatures and Blind Signatures". In: *J. Cryptology* 13.3 (2000), pp. 361–396 (cit. on pp. 50, 51, 98).

[69] J.M. Pollard. "Monte Carlo methods for index computation (mod p)". In: *Mathematics of computation* 32.143 (1978), pp. 918–924 (cit. on p. 40).

[70] J.J. Quisquater et al. "How to explain zero-knowledge protocols to your children". In: *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Springer, 1990, pp. 628–631. ISBN: 3-540-97317-6 (cit. on p. 24).

[71] R.L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126. ISSN: 0001-0782 (cit. on pp. 13, 14).

[72] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Springer, 1990, pp. 239–252. ISBN: 3-540-97317-6 (cit. on p. 34).

[73] Joseph H. Silverman. *Arithmetic of Elliptic Curves*. Third. Springer, 1986 (cit. on p. 38).

[74] Simon Singh. *The Code Book*. Fourth Estate, 2002 (cit. on p. 13).

[75] Nigel Smart. *Cryptography, an Introduction*. 3rd. 2010 (cit. on pp. 15, 19, 22, 36, 66).

[76] Nigel Smart. *ECRYPT Yearly Report on Algorithms and Keysizes (2010), D.SPA.13 Rev. 1.0*. `http://www.ecrypt.eu.org/documents/D.SPA.13-1.0.pdf`. Mar. 2010 (cit. on p. 40).

[77] Serge Vaudenay. “On privacy models for RFID”. In: *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*. Ed. by Kaoru Kurosawa. Vol. 4833. Lecture Notes in Computer Science. Springer, 2007, pp. 68–87. ISBN: 978-3-540-76899-9 (cit. on p. 57).

[78] Frederik Vercauteren. “Pairings on Elliptic Curves”. In: *Identity-Based Cryptography*. IOS Press, 2009. Chap. II (cit. on p. 41).

[79] Eric R. Verheul. “Self-Blindable Credential Certificates from the Weil Pairing”. In: *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*. Ed. by Colin Boyd. Vol. 2248. Lecture Notes in Computer Science. Springer, 2001, pp. 533–551. ISBN: 3-540-42987-5 (cit. on pp. 7, 9, 44, 83).

[80] Qingsong Ye, Huaxiong Wang, Josef Pieprzyk, and Xian-Mo Zhang. “Efficient Disjointness Tests for Private Datasets”. In: *Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings*. Ed. by Yi Mu, Willy Susilo, and Jennifer Seberry. Vol. 5107. Lecture Notes in Computer Science. Springer, 2008, pp. 155–169. ISBN: 978-3-540-69971-2 (cit. on pp. 79, 80).