UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Fakultät für Elektrotechnik, Informatik und Mathematik
Arbeitsgruppe Codes und Kryptographie

Anonymous Credential System based on the q-Strong Diffie-Hellman Assumption

Master's Thesis

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by
Fabian Eidens

submitted to

Prof. Dr. Johannes Blömer

and

Prof. Dr. Friedhelm Meyer auf der Heide

Paderborn, April 1, 2015

Contents

1 Introduction

Private information becomes increasingly accessible as more information is stored digitally and available via the Internet. Digitally stored user data such as purchase or search history is easy to analyze. Therefore it was never easier to create a comprehensive profile of a user. A system that let users control the degree of propagation of private information is needed. Techniques to protect private information and securing communication are ongoing studies in cryptography. These studies consider adversaries capable of intercepting and monitoring the communication between honest parties. One goal is that an adversary will not even learn a portion of the private information.

An application of private information protecting techniques is a credential system. A credential system consists of users, organizations and sometimes a trusted third party. The organizations or trusted third parties issue credentials to users. A user participating in a credential system, can get access to information only after proving the ownership of an authorizing credential. A physical world example for credential systems is the driver license, where the user's ownership of it proves that he is authorized to drive cars. Loosely speaking, to design a credential system in the digital world we issue credentials on the identity of users by creating a digital signature on it. A digital credential system has also to achieve several formal security properties such as consistency of credentials and anonymity of users. Consistency of credentials means that it is not possible to forge a credential for a user, this must hold even if all users and organizations work together.

The anonymity of users is the main property of anonymous credential systems (pseudonym systems). They were introduced by Chaum [21] and further studied in [22, 26, 23, 12, 35]. An anonymous credential system let users request credentials on pseudonyms and let them show the credentials to organizations without revealing any private information, such as the user's identity. Organizations know only the pseudonyms of a user and are not able to cooperatively link different pseudonyms of one user to identify him. Anonymous credential systems are needed if we think of a pharmacy that offer prescription drugs over the internet. A user has to show his prescription but he also want to keep his identity secret. Otherwise the pharmacy would be able to track his health statistic and perhaps sell this information to a third party. Using an anonymous credential system the user is able to anonymously prove that he has the prescription without revealing any private information. Another application of anonymous credential systems for electronic cash is presented in [34] by Lysyanskaya. With the use of the electronic cash system the user can also anonymously pay at the pharmacy.

1.1 State of research

To construct anonymous credential systems one can use general techniques of zero-knowledge proofs (of knowledge) and techniques of secure two-party computations as it is described Lysyanskaya in [34]. Lysyanskaya [34] also make clear that this approach is not efficient. A more efficient approach introduced by Camenisch and Lysyanskaya [34, 15] is to use a commitment scheme and a (digital) signature scheme, combined with efficient protocols. First a protocol for proving knowledge of a committed value. Second a protocol for proving knowledge of a signature on a committed value and third a protocol for signing a committed value without revealing the committed value to the signing party. In this context the commitment is the user's pseudonym. The commitment is then signed by an organization and represents the credential. In this model, if a user wants to obtain a credential from a organization A, he forms a commitment and follows the protocol for signing a committed value. In the protocol the user has to prove knowledge of the committed value. The end result is a credential for the pseudonym. The user is now able to prove the possession of a credential from organization A towards another organization B. In succession organization B grants the user access.

Commitment schemes like the Pedersen Commitment [38] let a user commit to a value towards a receiver. The value is only known to the user and hidden for any adversary and in particular for the receiver. If the user is asked by the receiver to reveal his committed value, the receiver is able to check the validity of the user's commitment.

Digital signature schemes were introduced by Diffie and Hellman [27]. They let users sign documents digitally as an equivalent of handwritten signatures. In general, digital signature schemes guarantee authenticity and integrity of documents. Additionally digital signatures are publicly verifiable and transferable. In a digital signature scheme a user has a unique pair of *secret key*, only available to the user, and a publicly available *verification key*. A message is signed using the *secret key* and can be verified by anyone using the *verification key*. Digital signature schemes were formalized by Goldwasser, Micali and Rivest [32] and they exist if and only if one-way functions exist [36, 41]. General constructions, that are based on one-way functions, lack efficiency and are therefore undesirable for practical use. More efficient signature schemes, like RSA [40], Fiat-Shamir [29] and the Schnorr [42] signature scheme, were shown to be secure in the random oracle model, where we assume that an ideal random function exist. The downside is that these schemes are not provable secure in the standard model.

Another efficient signature scheme that is secure in the random oracle model was constructed by Boneh, Boyen and Shacham in [8]. The security of their scheme is based on a discrete-logarithm-type assumption called q-Strong Diffie-Hellman (q-SDH) Assumption. Independently of [8], Camenisch and Lysyanskaya [15] proposed an efficient signature scheme that is provable secure in the standard model, where the security is based also on a discrete-logarithm-type assumption called LRSW Assumption. The assumption was introduced by Lysyanskaya, Rivest,

Sahai and Wolf in [35] and it was shown that it holds in generic groups. In [15] Camenisch and Lysyanskaya provide a set of efficient protocols to obtain a signature on a committed value and to prove knowledge of a signature on a committed value. Their main accomplishment is an anonymous credential system that is based on the LRSW Assumption. Other anonymous credential systems were proposed by Verheul [43] and Ateniese and de Medeiros [1]. The anonymous credential system of Verheul [43] is also based on a discrete-logarithm-type assumption but the system is not proven secure. Ateniese and de Medeiros [1] have proposed a system that is based on the Strong RSA Assumption. Another an anonymous credential system based on the Strong RSA Assumption is presented by Lysyanskaya in [34]. Lysyanskaya [34] also presents efficient protocols and a formal definition of secure anonymous credential systems.

1.2 Our Work

We design an anonymous credential system based on the q-SDH Assumption. For the design we rely on the definition of anonymous credential systems established by Lysyanskaya [34]. The idea for this anonymous credential system is originally presented by Camenisch and Lysyanskaya in [15]. They state that their methodology can be applied to the work of Boneh, Boyen and Shacham [8] to yield a q-SDH-based signature scheme with efficient protocols.

In this thesis we show signature schemes that are secure under the q-SDH Assumption and based on the mentioned idea. We also provide the corresponding efficient protocols. In combination with a commitment scheme for blocks of values we present an anonymous credential system based on the q-SDH Assumption.

In Chapter 2 we introduce notation, recall important definitions, show a commitment scheme for blocks of values and present basic protocols for the commitment scheme. Next, in Chapter 3 we recall the definition of secure anonymous credential systems established by Lysyanskaya [34]. We also present how the combination of a secure signature scheme, commitment scheme and efficient protocols result in a basic anonymous credential system. In Chapter 4 we present signature schemes that are secure under the q-SDH Assumption. Similar to Camenisch and Lysyanskaya [15], but under a different assumption, we present a basic signature scheme for single messages and a signature scheme for blocks of messages. Where the latter is suited for obtaining a signature on a committed value. In Chapter 5 we show efficient protocols which allow us to use our signature schemes in the design of an anonymous credential system. In detail, we present a protocol for signing a committed value and a protocol for proving knowledge of a signature on a committed value. Finally, in Chapter 6 we present our main result. The combination of the commitment scheme, signature scheme and efficient protocols in order to obtain an anonymous credential system based on the q-SDH Assumption.

2 Preliminaries

This chapter introduces the notation and definitions of, among others, digital signature schemes, commitment schemes, zero-knowledge proofs and proofs of knowledge. Further it introduces the complexity assumptions used in this thesis to design an anonymous credential system.

2.1 Notation

In this thesis we denote with $x \leftarrow S$ that x is chosen uniformly at random from a set S. A similar notation is used for a probabilistic polynomial-time (ppt) algorithm A. We denote with $x \leftarrow A$ that x is the output of A determined by the random choices of A. Further, we denote by $x \in A$ that the probability that x is the output of the ppt A is positive. A function $\nu\colon \mathbb{N} \to \mathbb{R}^+$ is called negligible, if for every $c \in \mathbb{N}$ there is a $\eta_0 \in \mathbb{N}$, such that for all $\eta \geq \eta_0$ it holds that $\nu(\eta) \leq 1/\eta^c$. Let $A(1^\eta)$ and $B(1^\eta)$ be two probabilistic algorithms. We call the outputs of A and B computational indistinguishable, if for all probabilistic polynomial-time (ppt) families of adversaries $\{\mathcal{A}_\eta\}$, there exists a negligible function ν such that for all η, $\Pr\left[x_0 \leftarrow A(1^\eta); x_1 \leftarrow B(1^\eta); b \leftarrow \{0,1\}\,; b' \leftarrow \mathcal{A}(x_b)\colon b = b'\right] = \nu(\eta)$. This notation is originally presented by Lysyanskaya [34]. By $A \overset{\blacksquare}{\to} B$ we denote that algorithm A has black-box access to algorithm B. Black-box access $A \overset{\blacksquare}{\to} B$ captures that A can invoke B with an input of his choice and reset the state of B. For an interactive proof system (protocol) between two Turing machines A and B we will use the following notation $a \leftarrow (A(x) \leftrightarrow B(x)) \to b$. Where a (resp. b) is the random variable representing the output of A (resp. B) when interacting with B (resp. A) on common input x.

By $A^O(\cdot)$ we denote a Turing machine that has oracle access to O. Queries to the oracle O are written in binary to an additional read/write-once query tape. Queries are concluded with the symbol #. The oracle O is invoked right after the conclusion of the query and the answer is written adjacent to the symbol #. In this thesis we assume that in the unlikely event that $x = 0 \mod p$ after choosing $x \leftarrow \mathbb{Z}_p$ it is repeated until $x \neq 0 \mod p$. We will sometimes use the notation for proofs of knowledge introduced by Camenisch and Stadler [16]. For example,

$$ZKPK\left\{(a,b,c)\colon A = g^a \wedge B = g^a \cdot h^b \wedge C = g'^{a+b} \cdot h'^c\right\}$$

denotes a zero-knowledge proof of knowledge of integers a, b and c such that $A = g^a$, $B = g^a \cdot h^b$ and $C = g'^{a+b} \cdot h'^c$ holds. Where g, h are generators of some

group $\mathbb{G}$ and g', h' are generators of some group $\mathbb{G}'$.

2.1.1 Bilinear Groups

Our signature schemes and anonymous credential system use bilinear groups. Therefore, we use a variant of the notation introduced in [9] by Boneh, Lynn and Shacham.

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are two (multiplicative) cyclic groups of prime order p
- Generator of $\mathbb{G}_1$ is g_1 ($\mathbb{G}_1 = \langle g_1 \rangle$) and of $\mathbb{G}_2$ is g_2 ($\mathbb{G}_2 = \langle g_2 \rangle$)
- ψ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$
- e is a computable map with $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, such that
  - (Bilinear) For all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, for all $a, b \in \mathbb{Z}$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$
  - (Non-degenerate) There exits $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity of $\mathbb{G}_T$

Definition 2.1 (Bilinear group pair)**.** We call $(\mathbb{G}_1, \mathbb{G}_2)$ a bilinear group pair if there exists a group $\mathbb{G}_T$ and a non-degenerate bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, such that the group order $p = |\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$ is prime, a computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$, and the pairing e and the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are all efficiently computable.

For the setup of bilinear group pairs we use bilinear group generators that are defined in the following.

Definition 2.2 (Bilinear group generator [7])**.** A bilinear group generator $\mathcal{G}$ is a probabilistic polynomial-time (ppt) algorithm that, on input 1^η, outputs prime p, the description of groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p. Further it outputs the description of an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ and a bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, such that $(\mathbb{G}_1, \mathbb{G}_2)$ form a bilinear group pair.

Notice, following Boneh, Boyen and Shacham [8] we allow that $\mathbb{G}_1 \neq \mathbb{G}_2$.

2.2 Complexity Assumptions

The following q-Strong Diffie-Hellmann Assumption was introduced by Boneh and Boyen in [6]. In the same paper they show how to construct a signature scheme whose security is based on the introduced assumption. We will use the assumption to prove our signature schemes secure and therefore to design an anonymous credential system.

2.2.1 Strong Diffie-Hellman Assumption

A variant of the following asymptotic definition of the q-Strong Diffie-Hellmann Assumption was originally presented by Boneh and Boyen in [7].

Definition 2.3 (q-Strong Diffie-Hellman Assumption - q-SDH [6, 7])**.** Let $\mathcal{G}$ be a bilinear group generator and $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$ such that $g_1 = \psi(g_2)$. The q-SDH problem in the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ is stated as follows.

Given a $(q+2)$-tuple of elements $(g_1, g_2, g_2^\gamma, g_2^{\gamma^2}, \ldots, g_2^{\gamma^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ as input, output a pair $\left(g_1^{\frac{1}{x+\gamma}}, x\right) \in \mathbb{G}_1 \times \mathbb{Z}_p$.

The q-SDH Assumption holds for $\mathcal{G}$ if, for every probabilistic polynomial-time (ppt) adversary $\mathcal{A}$ and for every polynomial bounded function $q\colon \mathbb{Z} \to \mathbb{Z}$ the probability,

$$\Pr\left[\mathcal{A}(g_1, g_2, g_2^\gamma, g_2^{\gamma^2}, \ldots, g_2^{\gamma^q}) = \left(g_1^{\frac{1}{x+\gamma}}, x\right) : g_2 \leftarrow \mathbb{G}_2,\ g_1 = \psi(g_2),\ \gamma \leftarrow \mathbb{Z}_p\right] = \nu(\eta)$$

is a negligible function in η. Where the probability is over the random uniform choices of the parameters g_2, γ and over the random bits used by $\mathcal{A}$.

2.2.2 Linear Diffie-Hellman Assumption

The following definition was presented in [8]. We will use it for the design of a protocol to prove the knowledge of a signature in Section 5.2.

Definition 2.4 (Decision Linear Problem in $\mathbb{G}_1$ - DLP)**.** Let $\mathbb{G}_1$ be a cyclic group of prime order p, with $\mathbb{G}_1 = \langle g_1 \rangle$. Given arbitrary generators $u, v, h, u^a, v^b, h^c \in \mathbb{G}_1$, where $a, b, c \leftarrow \mathbb{Z}_p$ output 1 if $c = a + b$ and 0 otherwise. The Decision Linear Problem (DLP) holds in $\mathbb{G}_1$ if for every probabilistic polynomial-time (ppt) adversary $\mathcal{A}$,

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = 1 : u, v, h \leftarrow \mathbb{G}_1, a, b \leftarrow \mathbb{Z}_p\right] \\ -\Pr\left[\mathcal{A}(u, v, h, u^a, v^b, h^c) = 1 : u, v, h \leftarrow \mathbb{G}_1, a, b, c \leftarrow \mathbb{Z}_p\right] \end{array} \right| = \nu(\eta)$$

is a negligible function in η. Where the probability is over the random choices of $\mathcal{A}$ and over the values u, v, h, a, b, c given to $\mathcal{A}$.

The following definition of Linear Encryption is presented in [8]. We use Linear Encryption as a part of our Protocol 5.9 for proving knowledge of a signature on a committed value.

Definition 2.5 (Linear Encryption - LE)**.** The public key is a triple of generators $u, v, h \in \mathbb{G}_1$. The secret key is pair of values $\xi_1, \xi_2 \in \mathbb{Z}_p$ such that $h^{\frac{1}{\xi_1}} = u$ and

$h^{\frac{1}{\xi_2}} = v$. An encryption of a message $M \in \mathbb{G}_1$ is formed as, choose random values $a, b \in \mathbb{Z}_p$ and output $(u^a, v^b, M \cdot h^{a+b})$. To decrypt a ciphertext (T_1, T_2, T_3), compute $M := T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2})$.

2.3 Digital Signature Schemes

A secure digital signature scheme is an essential part of the anonymous credential system that we will design. In such a system a credential corresponds to a signature. We will grant credentials by signing a committed value. Will describe the combination of signature schemes and credentials in the following chapters. First let us define digital signature schemes. The following definition of digital signature schemes is due to Goldwasser, Micali and Rivest [32].

Definition 2.6 (Signature Scheme)**.** A (digital) signature scheme Π is a triple of probabilistic polynomial-time (ppt) algorithms Gen, Sign and Verify where

1. $\mathrm{Gen}(1^\eta)$ outputs a key pair (pk, sk)
2. $\mathrm{Sign}(pk, sk, m)$ on input the secret key sk and a message $m \in \{0,1\}^*$ outputs a signature $\sigma \leftarrow \mathrm{Sign}(pk, sk, m)$.
3. $\mathrm{Verify}(pk, m, \sigma)$ where pk is the public key, $m \in \{0,1\}^*$ is a message and σ is a signature. Verify is deterministic and it outputs $b \in \{0,1\}$. Where $b = 1$ means valid and $b = 0$ means invalid.

Correctness: For every key pair (pk, sk) and message $m \in \{0,1\}^*$ it holds that

$$\mathrm{Verify}\,(pk, m, \mathrm{Sign}(pk, sk, m)) = 1$$

Definition 2.7 (Signature Forging Game Sig-forge$_{\mathcal{A},\Pi}(\eta)$)**.** Let $\mathcal{A}$ be an adversary and $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ a signature scheme. The game is defined as follows:

1. $(pk, sk) \leftarrow \mathrm{Gen}(1^\eta)$
2. $\mathcal{A}$ is given 1^η, pk and oracle access to $\mathrm{Sign}(pk, sk, \cdot)$. Let Q be the set of tuples containing the queries made by $\mathcal{A}$ to $\mathrm{Sign}(pk, sk, \cdot)$ and the corresponding answers of the signature oracle. Eventually $\mathcal{A}$ outputs a pair (m, σ).
3. The output is 1, iff
   a) $\mathrm{Verify}(pk, m, \sigma) = 1$,
   b) $(m, \sigma) \notin Q$.

Next, we define the security of a signature scheme using the game above. A signature scheme that satisfies the following definition is said to be secure against *adaptive chosen-message attacks*.

Definition 2.8 (Secure Signature Scheme)**.** Signature scheme Π is called existentially unforgeable under an adaptive chosen-message attack, or secure, if for every ppt adversary $\mathcal{A}$ there is a negligible function $\nu\colon \mathbb{N} \to \mathbb{R}^+$ such that

$$\Pr\left[\text{Sig-forge}_{\mathcal{A},\Pi}(\eta) = 1\right] \leq \nu(\eta)$$

2.4 Commitment Schemes

In an anonymous credential system we want to grant credentials (respectively signatures) on pseudonyms of users without telling the signer the real identities of the users. Therefore, we will only sign commitments. Where a user forms a pseudonym by using a provided commitment scheme. Let us first introduce an informal definition of a commitment scheme. Later we will formally define commitment schemes and its security.

There are various forms of commitment schemes. Its core always consists of a so called two-party protocol among a Committer and a Receiver. The protocol processes in two phases. The first phase is called *Commit* phase. The role of the Committer is to commit himself to a value during this phase. The output of this phase is a commitment. The second phase is called the *Reveal* phase. In this phase the committed value has to be revealed by the Committer towards the Receiver. The phase ends with the Receiver checking the validity of the commitment.

Loosely speaking, a commitment scheme is called secure if the following two properties hold. First, the Receiver learns nothing about the committed value. This property is called hiding. Second it has to hold that the committed value can not be exchanged by any (malicious) party after the *Commit* phase. This property is called binding. In the following we concentrate on non-interactive commitment schemes. The following definition is originally presented in [34] Section 2.3.

A non-interactive commitment scheme $\mathcal{C} = (\text{Gen}_{\mathcal{C}}, \text{Commit})$ consists of a setup algorithm $\text{Gen}_{\mathcal{C}}$ and a polynomial-time computable function Commit for the *Commit* phase. $\text{Gen}_{\mathcal{C}}$ on input the security parameter 1^η outputs the commitment public key pk. The public key also determines the message space $\mathcal{M}_{pk}$. Commit on input the public key pk, a message $x \in \mathcal{M}_{pk}$, and a random string r of η-bits outputs the commitment C. The two-party protocol starts with the *Commit* phase. The Committer picks a random r and computes the commitment to the value x as $C := \text{Commit}(pk, x, r)$. The *Commit* phase ends after the Committer has send C to the Receiver. In the *Reveal* phase the Receiver has to verify the commitment. Therefore, the values x and r are revealed by the Committer and the Receiver checks if $C = \text{Commit}(pk, x, r)$ holds. Let us now give a formal definition of non-interactive commitment schemes.

Definition 2.9 ((non-interactive) Commitment Scheme)**.** $\mathcal{C} = (\text{Gen}_{\mathcal{C}}, \text{Commit})$ is a non-interactive public-key commitment scheme if the key generation algorithm $\text{Gen}_{\mathcal{C}}$ runs in probabilistic polynomial-time and Commit is a polynomial-time

computable function that takes as input a public key, value x and random string r and outputs a commitment C such that the following two properties hold.

Perfect Hiding For every probabilistic polynomial-time adversary $\mathcal{A}$ that chooses two values x_0,$x_1 \in \mathcal{M}_{pk}$ and receives a commitment to a uniformly at random chosen one of them the following holds

$$\Pr\Big[pk \leftarrow \mathrm{Gen}_{\mathcal{C}}(1^\eta),\ (x_0, x_1) \leftarrow \mathcal{A}(1^\eta, pk),\ b \leftarrow \{0,1\}\,, \\ r \leftarrow \{0,1\}^\eta\,,\ C = \mathrm{Commit}(pk, x_b, r),\ b' \leftarrow \mathcal{A}(C)\colon b' = b\Big] \leq \frac{1}{2}$$

Computational Binding No probabilistic polynomial-time adversary $\mathcal{A}$ can open one commitment in two different ways. For all ppt algorithms $\mathcal{A}$, there exists a negligible function ε such that

$$\Pr\Big[pk \leftarrow \mathrm{Gen}_{\mathcal{C}}(1^\eta),\ (x_1, r_1, x_2, r_2) \leftarrow \mathcal{A}(1^\eta, pk)\colon x_1, x_2 \in \mathcal{M}_{pk} \wedge \\ x_1 \neq x_2 \wedge \mathrm{Commit}(pk, x_1, r_1) = \mathrm{Commit}(pk, x_2, r_2)\Big] = \varepsilon(\eta)$$

Notice that there are other definitions for hiding and binding. But in this thesis we only use perfect hiding and computational binding.

Next, we introduce the so called Pedersen Commitment [38]. After that we will extend it to support commitments to blocks of values. We will combine it with a signature scheme for blocks of messages to sign committed values. In terms of an anonymous credential system this corresponds to granting a credential to a pseudonym.

2.4.1 Pedersen Commitment

A standard commitment scheme is the so called Pedersen Commitment presented in [38]. Original the commitment scheme is defined as $\mathcal{C}_{Ped} = (\mathrm{Gen}_{\mathcal{C}}, \mathrm{Commit})$. Here $\mathrm{Gen}_{\mathcal{C}}$ on input 1^η outputs two generators g, h of $\mathbb{G}_1$ such that $\log_g(h)$ is not known. The Committer commits himself to a value $x \in \mathbb{Z}_p$ in the *Commit* phase by choosing $s \in \mathbb{Z}_p$ uniformly at random and computes $C := g^s h^x = \mathrm{Commit}(x, s)$. In the *Reveal* phase the committer reveals x and s. The Receiver can now check if $C = g^s h^x$ holds. This commitment scheme is known to be perfectly hiding and computational binding under the assumption that the discrete logarithm problem is hard in $\mathbb{G}_1$.

Next, we will present and prove the security of a natural extension of the Pedersen Commitment to blocks of values (messages).

2.4.2 Commitment Scheme for blocks

The following commitment scheme supports commitments to blocks of values. It is used in our protocol for obtaining a signature on a committed value where the signature should be created on a block of messages, see Protocol 5.1. Further, it is used in the protocol for proving knowledge of a signature on a committed value, see Protocol 5.9.

Definition 2.10. Let $\mathcal{G}$ be a bilinear group generator. The commitment scheme $\mathcal{C}^L = (\mathrm{Gen}_{\mathcal{C}}, \mathrm{Commit})$ is defined as follows:

$\mathrm{Gen}_{\mathcal{C}}(1^\eta)$:

1. Get $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$.
2. Choose $u_0, \ldots, u_L \leftarrow \mathbb{G}_2$
3. Set public key $pk := (u_0, \ldots, u_L)$

Commit phase:

1. To form a commitment to a block of values $M = (m_1, \ldots, m_L) \in \mathbb{Z}_p^L$, choose $s \leftarrow \mathbb{Z}_p$ and run $\mathrm{Commit}(pk, M, s)$ defined as:
   a) Output commitment $C := h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$ where $h_i := \psi(u_i)$ for $i = 0, \ldots, L$

Reveal phase:

1. The values $s, m_1, \ldots, m_L$ are revealed by the Committer to the Receiver.
2. The Receiver checks that $C = h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$ holds.

Theorem 2.11. *The commitment scheme of Definition 2.10 is a non-interactive commitment scheme that is computationally binding under the assumption that the discrete logarithm problem is hard in $\mathbb{G}_1$ and perfectly hiding.*

Proof. The theorem follows from the following Lemma 2.12 and Lemma 2.13. □

Lemma 2.12. *The commitment scheme $\mathcal{C}^L = (\mathrm{Gen}_{\mathcal{C}}, \mathrm{Commit})$ of Definition 2.10 is perfectly hiding.*

Proof. Let $C = h_0^s \cdot h_1^{m_1} \cdot h_2^{m_2} \cdot \ldots \cdot h_L^{m_L}$ be fixed. Then for every $M' = (m'_1, \ldots, m'_L)$ there is a $s' \in \mathbb{Z}_p$ such that $h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} = C = h_0^{s'} \cdot h_1^{m'_1} \cdot \ldots \cdot h_L^{m'_L}$ holds. The value s' can be computed as follows.

$$\begin{aligned} h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} &= h_0^{s'} \cdot h_1^{m'_1} \cdot \ldots \cdot h_L^{m'_L} \Leftrightarrow \\ h_0^s \cdot h_1^{m_1 - m'_1} \cdot \ldots \cdot h_L^{m_L - m'_L} &= h_0^{s'} \quad \Leftrightarrow \\ s + \alpha_1 (m_1 - m'_1) + \ldots + \alpha_L (m_L - m'_L) \mod p &= s' \end{aligned}$$

Where $\alpha_i \in \mathbb{Z}_p$ such that $h_0^{\alpha_i} = h_i$ for $i = 1, \ldots, L$. Since, there is such a value s' for every block of messages M' the following holds. For fixed but arbitrary $m = (m_1, \ldots, m_L) \in \mathbb{Z}_p^L$ and $C \in \mathbb{G}_1$ it holds that,

$$\begin{aligned}\Pr\left[C = \mathrm{Commit}(pk, s, M) | M = m\right] &= \Pr\left[C = h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} | M = m\right] \\ &= \Pr\left[C = h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}\right] \\ &= \Pr_{s \leftarrow \mathbb{Z}_p}\left[C = h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}\right] = \frac{1}{p} \qquad (2.1)\end{aligned}$$

We will now analyze the advantage of an adversary to distinguish two commitments. Let us look at the following game. For every probabilistic polynomial-time adversary $\mathcal{A}$,

1. The challenger chooses $pk \leftarrow \mathrm{Gen}_{\mathcal{C}}(1^\eta)$
2. $(M_0, M_1) \leftarrow \mathcal{A}(1^\eta, pk)$, where $M_0, M_1 \in \mathbb{Z}_p^L$
3. The challenger chooses $b \leftarrow \{0, 1\}$, $s \leftarrow \mathbb{Z}_p$ and computes $C = \mathrm{Commit}(pk, M_b, s)$
4. Adversary $\mathcal{A}$ on input C eventually outputs b'

We say that $\mathcal{A}$ wins the above game iff $b = b'$. Since Eq. (2.1) holds for all m, we have that for every probability distribution over $\mathbb{Z}_p$, every $M_0, M_1 \in \mathbb{Z}_p^L$ and every $C \in \mathbb{G}_1$.

$$\Pr\left[C = \mathrm{Commit}(pk, M, s) | M = M_0\right] = \frac{1}{p} = \Pr\left[C = \mathrm{Commit}(pk, M, s) | M = M_1\right]$$

Therefore the best that an arbitrary adversary $\mathcal{A}$ on input C can do is to guess with a (fair) coin.

$$\begin{aligned}\Pr\left[\mathcal{A} \text{ wins}\right] &= \Pr\left[b' \leftarrow \mathcal{A}(C) \wedge b = b' | b = 0\right] \cdot \Pr\left[b = 0\right] + \\ &\quad \Pr\left[b' \leftarrow \mathcal{A}(C) \wedge b = b' | b = 1\right] \cdot \Pr\left[b = 1\right] \\ &= \frac{1}{2} \cdot \Big(\Pr\left[b' \leftarrow \mathcal{A}(C) \wedge b = b' | b = 0\right] + \\ &\quad \Pr\left[b' \leftarrow \mathcal{A}(C) \wedge b = b' | b = 1\right] \Big) \\ &\leq \frac{1}{2}\end{aligned}$$

Which implies that the commitment scheme is perfectly hiding. □

Lemma 2.13. *The commitment scheme* $\mathcal{C}^L = (\mathrm{Gen}_{\mathcal{C}}, \mathrm{Commit})$ *of Definition 2.10 is computationally binding under the assumption that the discrete logarithm problem is hard in* $\mathbb{G}_1$.

An important part of an credential system is to be able to convince an organization that a user has a credential. At the same time we require that the organization does not learn the credential itself. Therefore we use zero-knowledge proofs of knowledge protocols. Let us first introduce the notion of zero-knowledge proofs.

2.5 Zero-Knowledge Proofs

Next we introduce zero-knowledge interactive proof systems. Informal, an interactive proof system is a communication between a prover P and a verifier V with a common input. The prover wants to convince the verifier that he has some kind of knowledge not available to the verifier. The zero-knowledge property captures that what we can efficiently compute after the interaction with prover P on common input x is the same as we were able to compute from x without the interaction. This is a property of the prover P. It captures P's robustness against attempts to gain knowledge by interacting with it [30, p. 200].

Definition 2.14 (Interactive Machine [30, p. 191])**.**

- An interactive Turing machine (ITM) is a (deterministic) multi-tape Turing machine. There are seven tapes. The tapes are a read-only *input tape*, a read-only *random tape*, a read-and-write *work tape*, a write-only *output tape*, a pair of *communication tapes*, and a read-and-write *switch tape* consisting of a single cell. One of the communication tapes is read-only and the other one is write-only.

- A single bit is associated to each ITM, called its identity. An ITM is said to be active, in a configuration, if the content of its switch tape is equal to the machine's identity. Otherwise the machine is said to be idle. In the idle state, the state of the machine including the locations of its heads and the contents of the writable tapes are preserved.

- The content of the input tape is called *input.* The content of the random tape is called *random input* and the content of the output tape at termination of the machine is called *output.* The content written on the write-only communication tape during a (time)-period in which the machine is active is called the *message sent* at that period. The content read from the read-only communication tape during an active period is called the *message received* at that period.

We will only look at a pair of machines combined together such that some of their tapes are shared. Two machines A and B can be combined in such a way that the read-only communication tape of B is the write-only communication tape of A or vice versa. Following [30, p. 191] the computation of such a pair of machines consists of the machines messages, based on their initial (common) input, their

(distinct) random inputs and the messages each machine has received up to the current period.

Definition 2.15 (Interactive Proof Systems [30, p. 199])**.**

- We denote by $(A(x,y) \leftrightarrow B(x,z)) \rightarrow \mathrm{b}$ the random variable representing the output of B when interacting with machine A on common input x, when the random input to each machine is uniformly and independently chosen. Where A (resp. B) has auxiliary input y (resp. z).
- A pair of interactive machines (P, V) is called an interactive proof system or interactive protocol for language L if V is polynomial-time and the following two conditions holds:
  - For every $x \in L$, there exists a string y such that for every $z \in \{0,1\}^*$,
  $$\Pr\left[P(x,y) \leftrightarrow V(x,z) \rightarrow \mathrm{b}\colon \mathrm{b}{=}1\right] \geq \frac{2}{3}$$
  - For every $x \notin L$, every interactive machine P^* and every $y, z \in \{0,1\}^*$,
  $$\Pr\left[P^*(x,y) \leftrightarrow V(x,z) \rightarrow \mathrm{b}\colon \mathrm{b}{=}1\right] \leq \frac{1}{3}$$
- Let $P_L(x)$ be the set of strings y satisfying the first property with respect to $x \in L$.

Such auxiliary inputs are for example used when the interactive proof system is a sub-protocol of another larger protocol. In the following we will refer to P as the prover and to V as the verifier. If a protocol is designed to be used in practice then interactive arguments are considered where the prover P is also limited to probabilistic polynomial-time. In this case the prover's capability comes from his auxiliary input.

Let us now define transcripts. Loosely speaking, they capture the input and corresponding output of an interactive proof system (protocol). After that we will define zero-knowledge protocols using transcripts.

Definition 2.16 (Transcripts [30])**.** Let (P, V) be an interactive proof system for some language L and $x{\in}L$. A transcript $\tau \in \{0,1\}^*$ of (P, V) on (common) input x consists of the input x, auxiliary input $y \in P_L(x)$, $z \in \{0,1\}^*$, the output of (P, V) and all messages exchanged between V and P. We denote by $T\left(P(x,w) \leftrightarrow V(x,z)\right)$ a random variable describing the transcript of the protocol executed by P and V. By $\Pr\left[T\left(P(x,w) \leftrightarrow V(x,z)\right) = \tau\right]$ we denote the probability that the transcript of (P, V) on (common) input x is equal to τ.

Similarly we denote by $S(v)$ for a probabilistic polynomial-time algorithm S the random variable corresponding to the output of S on input v. Further, by $\Pr\left[S(v) = \tau\right]$ we denote the probability that S on input v outputs τ.

Definition 2.17 (Zero-Knowledge (ZK) [30])**.** Let (P, V) be an interactive proof system for some language L. (P, V) is called zero-knowledge if for every probabilistic polynomial-time interactive machine (verifier) V^* there exists a probabilistic polynomial-time algorithm S (simulator) such that for all $x \in L$, all $w \in P_L(x)$, $z \in \{0,1\}^*$ and $\tau \in \{0,1\}^*$,

1. with probability $\leq \frac{1}{2}$, S outputs a special symbol $\perp$.
2. $\Pr\left[T\left(P(x,w) \leftrightarrow V^*(x,z)\right) = \tau\right] = \Pr\left[S(x,z) = \tau \mid S(x,z) \neq \perp\right]$

Notice that V^* may behave differently from V in the protocol $P(x,w) \leftrightarrow V^*(x,z)$, but P behaves as in $P(x,w) \leftrightarrow V(x,z)$.

Definition 2.18 (Honest-Verifier Zero-Knowledge (HVZK))**.** Let (P, V) be an interactive proof system for some language L. (P, V) is called honest-verifier zero-knowledge if there exists a probabilistic polynomial-time algorithm S (simulator) such that for all $x \in L$, all $w \in P_L(x)$, $z \in \{0,1\}^*$ and $\tau \in \{0,1\}^*$ the following holds,

$$\Pr\left[T\left(P(x,w) \leftrightarrow V(x,z)\right) = \tau\right] = \Pr\left[S(x,z) = \tau\right]$$

This is also called *perfect* honest-verifier zero-knowledge. To formulate *computational* honest-verifier zero-knowledge we require that the probability distributions on the transcripts of the real protocol $(P(\cdot,\cdot) \leftrightarrow V(\cdot,\cdot))$ and the simulation are only computationally indistinguishable.

2.6 Proofs of Knowledge

In general we want that a user is able to convince another party of the fact that he knows a secret. In an anonymous credential system we will use for this purpose zero-knowledge proofs of knowledge. For example we want to convince an organization that a user knows the opening of a commitment. More formally, if a prover has some kind of knowledge this knowledge has to be extractable in some way. The notion of knowledge extraction is captured by the so called extractor algorithm. We use a variant of the definitions presented by Lysyanskaya [34]. Let us first define computable binary relations. They are later used to define proofs of knowledge and Σ-protocols.

Definition 2.19 (Relation [30, p. 240, 254])**.** R is a polynomial-time computable binary relation if the following holds:

- There exists a polynomial $q(\cdot)$ such that for every $(x,w) \in R$, it holds that $|w| \leq q(|x|)$.
- There exists a polynomial-time algorithm for deciding membership in R.

A proof of knowledge can also be interpreted as a protocol whereby a prover convinces a verifier that he knows a quantity w that satisfies some relation R. For example, w (witness) can be a satisfying assignment to a boolean formula ϕ ($R(\phi, w) = \phi(w)$). In the following the witness will only be known to the prover. Notice that we denote by $(P(x, w) \leftrightarrow V(x)) \rightarrow b$ the random variable representing the output of V when interacting with P on common input x and auxiliary input w to P.

Definition 2.20 (Proof of Knowledge [34])**.** Let $R(\cdot, \cdot)$ be a polynomial-time computable relation. Let $u(x)$ be such that $|w| \leq u(x)$ for all w such that $R(x, w)$ holds, and assume that some such $u(x) = poly(|x|)$ is efficiently computable. A verifier V is a knowledge verifier with respect to R if:

Non-triviality There exists a prover P such that for all x, w, if $R(x, w) = 1$, then

$$\Pr\left[(P(x, w) \leftrightarrow V(x)) \rightarrow b \colon b = 1\right] = 1$$

Extraction with knowledge error $2^{-u(x)}$ There exists an extractor algorithm $\mathcal{E}$ and a constant c such that for all x, for all adversaries $\mathcal{A}$, if

$$p(x) = \Pr\left[(\mathcal{A} \leftrightarrow V(x)) \rightarrow b \colon b = 1\right] > 2^{-u(x)}$$

then, on input x and with (rewindable black-box) access to the prover, $\mathcal{E}$ computes a value w such that $R(x, w)$ holds, within an expected number of steps bounded by $\frac{(|x|+u(x))^c}{p(x)-2^{-u(x)}}$.

V is a verifier with respect to language L if V is a verifier with respect to the relation $R(x, w)$. Where $R(x, w) = 1$ if and only if w is a witness to the statement $x \in L$. Relation R is then also called witness relation.

Definition 2.21 (Witness Relation)**.** Let R_L be a polynomial-time computable binary relation. R_L is called a witness relation for language L if it holds that $L = \{x \colon \exists w \text{ s.t. } (x, w) \in R_L\}$.

A value w is called *witness* for $x \in L$, if $(x, w) \in R_L$ holds.

One of the *building* blocks of our anonymous credential system are efficient protocols. We will only define so called Σ-protocols in this thesis. Further, we will use a standard technique by Damgård [24] to convert Σ-protocols to zero-knowledge proofs of knowledge. First let us start with the definition of Σ-protocols.

Definition 2.22 (Σ-protocol [25])**.** Let (P, V) be an interactive proof system. (P, V) is said to be a Σ-protocol for witness relation R_L if:

- (P, V) is a three-round (interactive) proof system for language L, where V is a verifier with respect to L. (P, V) starts with a message y from P to V. Next V's only message to P (the challenge c) consists of its random coins and the last message r is from P to V.

- *Completeness*: If P and V follow the protocol on (common) input x and P's private input w, where $(x, w) \in R$, the verifier always accepts.

- From any x and any pair of accepting transcripts (y, c, r), (y, c', r') on input x, where $c \neq c'$, one can efficiently compute w such that $(x, w) \in R$. This is also called the *special soundness* property.

- There exists a polynomial-time simulator S, which on input $x \in L$ and a random c'' outputs an accepting transcript of the form (y'', c'', r'') with the same probability distribution as transcripts of the protocol between honest P, V on input x. This is also called *special honest-verifier zero-knowledge* (SHVZK).

Theorem 2.23 ([25])**.** *Let (P, V) be a* Σ-protocol *for witness relation R_L with challenge length t. Then (P, V) is a proof of knowledge with knowledge error 2^{-t}.*

Proof. The proof is presented in full detail in [25] and left out here. □

The following theorem is due to Damgård [24]. We will refer to it in our protocols as a standard technique to transform Σ-protocols to general zero-knowledge proofs of knowledge.

Theorem 2.24. *Given any witness relation R_L and a* Σ-protocol *for R_L. If one-way functions exists, then there are zero-knowledge three-round proofs of knowledge for R_L.*

2.7 Protocols

The following protocols are designed to yield efficient protocols for our signature schemes presented in Chapter 4. In fact what we will show here are protocols that we will use as parts of other protocols in Chapter 5. Besides that they are of interest on their own. For example for our commitment scheme and other applications. We will first present a protocol for proving knowledge of a committed value. Second, we will present a protocol for proving the equality of committed values. The second protocol is based on the first one. The involved commitments are of the commitment scheme $\mathcal{C}^L = (\mathrm{Gen}_{\mathcal{C}}, \mathrm{Commit})$ from Definition 2.10. In the following we will rely on Σ-protocols for the construction of our protocols. As mentioned before, with a standard technique by Damgård [24] we can transform our Σ-protocols to general zero-knowledge proofs of knowledge. We assume that the groups are set up by some party, that is not necessarily involved in one of the protocols.

2.7.1 Protocol for Proving Knowledge of a Committed Value

The following protocol is a variant of the Schnorr Protocol [42] and Okamoto Protocol [37]. Brands [11] shows how to do rapid demonstrations based on a similar protocol. The following protocol is a (P, V) interactive proof system for relation R_L. The relation R_L consists of tuples $(H, (m_0, \ldots, m_L)) \in \mathbb{G}_1 \times \mathbb{Z}_p^{L+1}$, such that $H = h_0^{m_0} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$.

Protocol 2.25. *Let the public parameters be $\mathbb{G}_1$ a cyclic group with prime order p, $h_i \in \mathbb{G}_1$, h_i generator for $i = 0, \ldots, L$, value L and $H = h_0^{m_0} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$. Let $m_0, \ldots, m_L \in \mathbb{Z}_p$ be the witness. The protocol steps are defined as follows.*

1. *P chooses $k_0, \ldots, k_L \leftarrow \mathbb{Z}_p$.*
2. *P computes $Y = \prod_{i=0}^{L} h_i^{k_i}$ and sends Y to V.*
3. *V chooses a challenge $c \leftarrow \mathbb{Z}_p$ and sends it to P.*
4. *P sets $r_i = k_i + c \cdot m_i$ for all $i = 0, \ldots, L$ and sends $r_0, \ldots r_L$ to V.*
5. *V accepts iff $Y = H^{-c} \prod_{i=0}^{L} h_i^{r_i}$ holds.*

Next we will prove that the protocol is complete. After that we will show in two lemmata that it is a Σ-protocol.

Lemma 2.26. *Protocol 2.25 is complete (i.e. an interaction with an honest prover is always accepted by the verifier).*

Proof. If prover P and verifier V follow the protocol as defined, we get for $i = 0, \ldots, L$, $r_i = k_i + c \cdot m_i$ and $Y = \prod_{i=0}^{L} h_i^{k_i}$ the following,

$$\begin{aligned} H^{-c} \prod_{i=0}^{L} h_i^{r_i} &= h_0^{-cm_0} \cdot \ldots \cdot h_L^{-cm_L} \cdot h_0^{k_0+cm_0} \cdot \ldots \cdot h_L^{k_L+cm_L} \\ &= h_0^{k_0} \cdot \ldots \cdot h_L^{k_L} \\ &= Y \end{aligned}$$

The check holds. Therefore, verifier V will always accept if prover P is honest. □

Lemma 2.27. *Protocol 2.25 is a honest-verifier zero-knowledge (HVZK) protocol under the assumption that the discrete logarithm problem is hard in $\mathbb{G}_1$.*

Proof. Simulator S on input public parameters $\mathbb{G}_1$ a cyclic group with prime order p, $h_i \in \mathbb{G}_1$, h_i generator for $i = 0, \ldots, L$, value L and $H' = h_0^{m'_0} \cdot \ldots \cdot h_L^{m'_L}$ proceeds as follows:

1. $c' \leftarrow \mathbb{Z}_p$

2. $r'_i \leftarrow \mathbb{Z}_p$ for $i = 0, \ldots, L$
3. $Y' = H'^{-c'} \cdot \prod_{i=0}^{L} h_i^{r'_i}$

Observe, that the verification check directly holds. Let the produced transcript of S be (c', r'_i, Y') for $i = 0, \ldots, L$. The probability of the transcript is $\frac{1}{p} \cdot \frac{1}{p^{L+1}}$. This holds since c' and $r'_0, \ldots, r'_L$ is chosen uniformly at random from $\mathbb{Z}_p$. The same probability holds for the transcripts in the real protocol. There we have that $r_0, \ldots r_L$ is fixed after $k_0, \ldots, k_L$ and c were chosen uniformly at random from $\mathbb{Z}_p$. It follows that the probability distribution of the transcripts of the real protocol and S are equal. □

Protocol 2.25 is also *special honest-verifier zero-knowledge* (SHVZK). We give the simulator S as an additional input $c' \leftarrow \mathbb{Z}_p$. Therefore, the fist step of S is omitted. The resulting probability distribution of the transcripts conditioned on c' in the simulation and in the real protocol is the same. Hence, Protocol 2.25 is a SHVZK protocol.

Lemma 2.28. *Protocol 2.25 satisfies the special soundness property.*

Proof. Suppose an algorithm $\mathcal{E}$ that is given two accepting transcripts of Protocol 2.25. Let $H = h_0^{m_0} \cdot h_1^{m_1} \cdot h_2^{m_2} \cdot \ldots \cdot h_L^{m_L}$, $H \in \mathbb{G}_1$ and let $(c, r_0, \ldots, r_L, Y)$ and $(c', r'_0, \ldots, r'_L, Y)$ be two accepting transcripts, where $c - c' \neq 0 \mod p$. B uses $H^{-c} \cdot \prod_{i=0}^{L} h_i^{r_i} = Y = H^{-c'} \cdot \prod_{i=0}^{L} h_i^{r'_i}$ to compute a witness as follows:

$$
\begin{aligned}
H^{c-c'} &= \prod_{i=0}^{L} h_i^{r_i} \cdot \prod_{i=0}^{L} h_i^{-r'_i} && \Leftrightarrow \\
H^{c-c'} &= \prod_{i=0}^{L} h_i^{r_i - r'_i} && \Leftrightarrow \\
H &= \prod_{i=0}^{L} h_i^{\frac{r_i - r'_i}{c-c'}}
\end{aligned}
$$

Hence $\mathcal{E}$ is able to extract (respectively compute) a witness $\left(\frac{r_i - r'_i}{c-c'}\right)$ for $i = 0, \ldots, L$. Notice that B runs in polynomial-time, since there it only divides two instances of the verification equation. Notice that the witness is perforce the same as in the protocol, since $\frac{r_i - r'_i}{c-c'} = \frac{k_i + cm_i - k_i + c'm_i}{c-c'} = \frac{m_i(c-c')}{c-c'} = m_i \mod p$ for $i = 0, \ldots, L$. □

Theorem 2.29. *Protocol 2.25 is a* Σ-protocol *for proving the knowledge of the values* $m_0, \ldots, m_L \in \mathbb{Z}_p$ *such that* $H = h_0^{m_0} \cdot \ldots \cdot h_L^{m_L}$.

Proof. The theorem follows from the completeness (Lemma 2.26), (S)HVZK property (Lemma 2.27), special soundness property (Lemma 2.28) and the three-round form of the protocol.

As mentioned before the next protocol is based on the last one and is used to prove the equality of committed values regarding two different commitments that are formed using different bases. □

2.7.2 Protocol for Proving the Equality of Committed Values

The next protocol is a Σ-protocol for proving knowledge of the values $s_1, s_2, m_1, \ldots, m_L \in \mathbb{Z}_p$ such that $C_1 = h_0^{s_1} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} \wedge C_2 = d_0^{s_2} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L}$. Where C_1 and C_2 are commitments that the commitment scheme of Definition 2.10 produces. Therefore, the commitments are computational binding under the assumption that the discrete logarithm problem is hard in $\mathbb{G}_1$ and perfectly hiding. Let us first define the protocol and then show that it is a Σ-protocol.

Protocol 2.30 (Protocol for proving the equality of committed values)**.** *Let the public parameters be $\mathbb{G}_1$ a cyclic group with prime order p, $h_i, d_i \in \mathbb{G}_1$, h_i, d_i generator for $i = 0, \ldots, L$, value L, $C_1 = h_0^{s_1} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$ and $C_2 = d_0^{s_2} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L}$. Let $s_1, s_2, m_1, \ldots, m_L \in \mathbb{Z}_p$ be the secret input (witness) to prover P. The protocol steps for prover P and verifier V are defined as follows.*

1. *P chooses $k_1, \ldots, k_L, o_1, o_2 \leftarrow \mathbb{Z}_p$.*
2. *P computes $Y_1 = h_0^{o_1} \cdot \prod_{i=1}^{L} h_i^{k_i}$, $Y_2 = d_0^{o_2} \prod_{i=1}^{L} d_i^{k_i}$ and sends Y_1, Y_2 to V.*
3. *V chooses a challenge $c \leftarrow \mathbb{Z}_p$ and sends it to P.*
4. *P sets $r_i = k_i + c \cdot m_i$ for $i = 1, \ldots, L$, $r_{o_1} = o_1 + c \cdot s_1$, $r_{o_2} = o_2 + c \cdot s_2$ and sends $r_1, \ldots r_L, r_{o_1}, r_{o_2}$ to V.*
5. *V accepts iff $Y_1 = C_1^{-c} \cdot h_0^{r_{o_1}} \cdot \prod_{i=1}^{L} h_i^{r_i}$ and $Y_2 = C_2^{-c} \cdot d_0^{r_{o_2}} \cdot \prod_{i=1}^{L} d_i^{r_i}$ holds.*

Next we will show that the protocol is complete. After that, we will show that the protocol is a Σ-protocol. Therefore, we will first prove that it is a special honest-verifier zero-knowledge (SHVZK) protocol. Then we will prove that it satisfies the special soundness property.

Lemma 2.31. *Protocol 2.30 is complete.*

Proof. If prover P and verifier V follow the protocol as defined, it holds that $Y_1 = h_0^{o_1} \cdot \prod_{i=1}^{L} h_i^{k_i}$, $Y_2 = d_0^{o_2} \prod_{i=1}^{L} d_i^{k_i}$ and $r_i = k_i + c \cdot m_i$ for $i = 1, \ldots, L$, $r_{o_1} = o_1 + c \cdot s_1$, $r_{o_2} = o_2 + c \cdot s_2$. Hence, the following hold:

$$\begin{aligned} C_1^{-c} \cdot h_0^{r_{o_1}} \cdot \prod_{i=0}^{L} h_i^{r_i} &= h_0^{-cs_1} \cdot h_1^{-cm_1} \cdot \ldots \cdot h_L^{-cm_L} \cdot h_0^{o_1+cs_1} \cdot h_1^{k_1+cm_1} \cdot \ldots \cdot h_L^{k_L+cm_L} \\ &= h_0^{o_1} \cdot h_1^{k_1} \cdot \ldots \cdot h_L^{k_L} \\ &= Y_1 \end{aligned}$$

and

$$\begin{aligned} C_2^{-c} \cdot d_0^{r_{o_2}} \cdot \prod_{i=0}^{L} d_i^{r_i} &= d_0^{-cs_2} \cdot d_1^{-cm_1} \cdot \ldots \cdot d_L^{-cm_L} \cdot d_0^{o_2+cs_2} \cdot d_1^{k_1+cm_1} \cdot \ldots \cdot d_L^{k_L+cm_L} \\ &= d_0^{o_2} \cdot d_1^{k_1} \cdot \ldots \cdot d_L^{k_L} \\ &= Y_2 \end{aligned}$$

Both checks hold. Therefore, verifier V will always accept if prover P is honest. □

Lemma 2.32. *Protocol 2.30 is a special honest-verifier zero-knowledge (SHVZK) protocol under the assumption that the discrete logarithm problem is hard in* $\mathbb{G}_1$.

Proof. Simulator S on input public parameters $\mathbb{G}_1$ a cyclic group with prime order p, $h_i, d_i \in \mathbb{G}_1$, h_i, d_i generator for $i = 0, \ldots, L$, value L, $C_1' = h_0^{s_1} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$, $C_2' = d_0^{s_2} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L}$ and challenge c' proceeds as follows:

1. $r'_{o_1}, r'_{o_2}, r'_i \leftarrow \mathbb{Z}_p$ for $i = 1 \ldots, L$
2. $Y_1' = C_1'^{-c'} \cdot h_0^{r'_{o_1}} \cdot \prod_{i=1}^{L} h_i^{r'_i}$
3. $Y_2' = C_2'^{-c'} \cdot h_0^{r'_{o_2}} \cdot \prod_{i=1}^{L} h_i^{r'_i}$

With the values set as defined above the V's check is directly satisfied. Let the produced transcript of S be $(r'_{o_1}, r'_{o_2}, r'_i, Y_1', Y_2')$ for $i = 1, \ldots, L$. We will analyze the output probability of the transcript for the given challenge c'. Since, c' is fixed in advance and $r'_{o_1}, r'_{o_2}, r'_1, \ldots, r'_L$ is chosen uniformly at random from $\mathbb{Z}_p$ we have that the probability for the above transcript is $\frac{1}{p^{L+2}}$. The same probability holds for the transcripts in the real protocol. Suppose we fix $c = c'$. Then we have that the values $r_{o_1}, r_{o_2}, r_1, \ldots r_L$ are fixed after $k_1, \ldots, k_L, o_1, o_2$ were chosen uniformly at random from $\mathbb{Z}_p$. It follows that for given c' the probability distribution of the transcripts of the real protocol and S are equal. □

Lemma 2.33. *Protocol 2.30 satisfies the special soundness property.*

Proof. Let $\mathcal{E}$ be an algorithm with input $C_1 = h_0^{m_0} \cdot h_1^{m_1} \cdot h_2^{m_2} \cdot \ldots \cdot h_L^{m_L}$, $C_2 = d_0^{s_2} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L}$ $C_1, C_2 \in \mathbb{G}_1$, two accepting transcripts $(c', r'_{o_1}, r'_{o_2}, r'_i, Y_1, Y_2)$, $(c, r_{o_1}, r_{o_2}, r_i, Y_1, Y_2)$ for $i = 1, \ldots, L$ and where $c - c' \neq 0 \mod p$. Since they are accepting transcripts we have that $Y_1 = C_1^{-c} \cdot h_0^{r_{o_1}} \cdot \prod_{i=1}^{L} h_i^{r_i} = C_1^{-c'} \cdot h_0^{r'_{o_1}} \cdot \prod_{i=1}^{L} h_i^{r'_i}$

and $Y_2 = C_2^{-c} \cdot d_0^{r_{o_2}} \cdot \prod_{i=1}^{L} d_i^{r_i} = C_2^{-c'} \cdot d_0^{r'_{o_2}} \cdot \prod_{i=1}^{L} d_i^{r'_i}$. Therefore,

$$\begin{aligned} C_1^{c-c'} &= h_0^{r_{o_1}} \cdot \prod_{i=1}^{L} h_i^{r_i} \cdot h_0^{-r'_{o_1}} \cdot \prod_{i=1}^{L} h_i^{-r'_i} && \Leftrightarrow \\ C_1^{c-c'} &= h_0^{r_{o_1}-r'_{o_1}} \cdot \prod_{i=1}^{L} h_i^{r_i-r'_i} && \Leftrightarrow \\ C_1 &= h_0^{\frac{r_{o_1}-r'_{o_1}}{c-c'}} \cdot \prod_{i=1}^{L} h_i^{\frac{r_i-r'_i}{c-c'}} \end{aligned}$$

and

$$\begin{aligned} C_2^{c-c'} &= d_0^{r_{o_2}} \cdot \prod_{i=1}^{L} d_i^{r_i} \cdot d_0^{-r'_{o_2}} \cdot \prod_{i=1}^{L} d_i^{-r'_i} && \Leftrightarrow \\ C_2^{c-c'} &= d_0^{r_{o_2}-r'_{o_2}} \cdot \prod_{i=1}^{L} d_i^{r_i-r'_i} && \Leftrightarrow \\ C_2 &= d_0^{\frac{r_{o_2}-r'_{o_2}}{c-c'}} \cdot \prod_{i=1}^{L} d_i^{\frac{r_i-r'_i}{c-c'}} \end{aligned}$$

Hence $\mathcal{E}$ is able to extract (resp. compute) a witness $\left(\frac{r_{o_1}-r'_{o_1}}{c-c'}, \frac{r_{o_2}-r'_{o_2}}{c-c'}, \frac{r_i-r'_i}{c-c'}\right)$ for $i = 1, \ldots, L$. Overall $\mathcal{E}$ runs in polynomial-time, since the only thing it does is the division of the two instances of the verification equation. Notice that the witness is perforce the same as in the protocol, since $\frac{r_i-r'_i}{c-c'} = \frac{k_i+cm_i-k_i+c'm_i}{c-c'} = \frac{m_i(c-c')}{c-c'} = m_i \mod p$ for $i = 1, \ldots, L$, $\frac{r_{o_1}-r'_{o_1}}{c-c'} = \frac{o_1+cs_1-o_1-c's_1}{c-c'} = \frac{s_1(c-c')}{c-c'} = s_1 \mod p$ and $\frac{r_{o_2}-r'_{o_2}}{c-c'} = \frac{o_2+cs_2-o_2-c's_2}{c-c'} = \frac{s_2(c-c')}{c-c'} = s_2 \mod p$. □

Theorem 2.34. *Protocol 2.30 is a* Σ-protocol *for proving knowledge of the values* $s_1, s_2, m_1, \ldots, m_L \in \mathbb{Z}_p$ *such that* $C_1 = h_0^{s_1} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} \wedge C_2 = d_0^{s_2} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L}$.

Proof. The theorem follows from the completeness (Lemma 2.31), SHVZK property (Lemma 2.32), special soundness property (Lemma 2.33) and the three-round form of the protocol. □

Notice, to use Protocol 2.25, Protocol 2.30 and any other Σ-protocol in our design of an anonymous credential system we have to convert it to a general zero-knowledge proof of knowledge using the standard technique by Damgård [24].

2.8 Two-Party Computations

In this section we will introduce the idea of two-party computations. This section motivates the background of the two-party computation that will be used in the

framework for anonymous credential systems in Chapter 3.

Two-party computations are also called two-party protocols. To give an informal description of secure computation, let g be a two-argument function $g\colon S_1 \times S_2 \to S_3$ with S_1, S_2 and S_3 finite sets. In the model two possibly malicious parties A and B are considered. A has a secret input $x \in S_1$ and B a secret input $y \in S_2$. Loosely speaking A and B exchange messages to securely compute $z = g(x, y)$. Secure computations mean that the protocol satisfies a correctness and a privacy property. Correctness means that the value computed by the protocol is equal to z. Privacy captures that A and B do not learn more about the other parties input as they should. In a restricted model it is required that they learn nothing about the others input. The above informal description is based on the introduction of [4].

A formal definition for general two-party computations was originally presented in [31, Chapter 7]. This definition aims at non-trivial functions that cannot be securely computed if a cheating party has unlimited computational power. They are based on functions that are computed as a Boolean circuit. This approach requires expensive computations such as a gate by gate evaluation of a Boolean circuit.

We will use two-party computations in the design of anonymous credential systems. For example, we want to obtain a signature on a committed value and prove a statement as "I have a signature". We will use a protocol for signing a committed value in the design of our anonymous credential system. One requirement of such a protocol is that the user's committed value and on the other side the signing key of the other party should be kept secret.

To design efficient systems we need efficient protocols. Camenisch and Lysyanskaya [15, 34] proposed the design of signature schemes that will easily yield itself to the construction of efficient protocols. Overall, proving a statement as "I have a signature" should be a more efficient process than representing this statement as a circuit and proving something about that circuit [34]. In this thesis we will present signature schemes and corresponding efficient protocols. First, a protocol for signing a committed value without revealing the value to the signing party. Second, a protocol for proving knowledge of a signature on a committed value. The notion of an anonymous credential system, its functional specification and security definition are presented in the next chapter.

3 Anonymous Credential Systems

Let us first give a short motivation for anonymous credential systems. Protecting the privacy of individuals is a challenging, but also very important task as more private information gets shared and analyzed. Many concerns of the society, but also of enterprises and organizations, involves the lack of control and protection of private information. A practical example are services that allow users to login using login data (resp. credentials) of other services. During this process crucial private information is shared among the services like name, age, gender and location. The point is that the majority of services do not need private information to grant users access. The only thing they need, and therefore should demand, is the knowledge that the user owns such a credential. Therefore, to leave out private information we should use anonymous credential systems.

An anonymous credential system let users obtain credentials from organizations (resp. services) and let them prove the knowledge (possession) of the credentials to another organization (resp. service). To protect the identity of the users, they form pseudonyms. Thus, in an anonymous credential system organizations identify the users only by pseudonyms. Therefore, organizations grant credentials only to pseudonyms. The user can then prove the possession of this credential to another organization, who knows the user under a different pseudonym. In this process the corresponding user only reveals that he owns such a credential. There are credential systems that support unlimited use credentials (*multiple-show* credentials) and one-time use credentials (*one-show* credentials) [35].

This chapter shows in detail how an anonymous credential system can be designed using a secure signature scheme, a commitment scheme and efficient protocols based on the work of Camenisch and Lysyanskaya [34, 15]. The definitions for credential systems are originally presented by Lysyanskaya [34, Chapter 3]. The definitions themselves are based on the work on multi-party computations by Canetti [17, 18] and the work on reactive systems by Pfitzmann and Waidner [39]. First we will describe the properties of an anonymous credential system. After that, the security for an anonymous credential system will be defined as follows. We will present a definition of a cryptographic system (CS) and its security in regard to an ideal-world specification (IS). After that we will show an anonymous credential system realized as a ideal-world specification and as a cryptographic system. Where the latter conforms to the IS and is said to be secure.

3.1 Properties of Credential Systems

The basic desirable properties of a credential system, like anonymity of users, are presented in the following. Further, we will present additional desirable properties concerning the problem of users sharing their pseudonyms and credentials.

3.1.1 Basic Properties

Our basic anonymous credential system will satisfy the following basic properties originally presented in [34, Chapter 3].

Unforgeability It is impossible to forge a credential for a user. This is even true if all users and organizations work together and run an adaptive attack on the credential issuing organization.

Consistency of credentials It is not possible that two or more users team up and show a set of their credentials to an organization and obtain a credential for one of the users that this user alone would not have gotten.

Autonomy of organizations An organization is able to choose his own cryptographic keys independent of the other parties of the system.

Anonymity of users An organization learns nothing apart from the fact that a user possess some set of credentials. This holds even if the organizations cooperate.

Unlinkability Two arbitrary pseudonyms of one and the same user cannot be linked. Furthermore, credential showings cannot be linked.

Efficiency Each interaction involves as few parties as possible. The communication rounds and communicated data are minimal.

Regarding the efficiency, our basic anonymous credential system in the cryptographic system model will have interactions between two parties, i.e. a user and an organization (resp. verifier).

3.1.2 Additional Properties

Additional properties are in some cases easy to add, for example if a public-key infrastructure is available. We will describe the properties and their advantages in the following. They are an abstract of the properties presented in [34, 35].

Sharing A User forms a pseudonym (commitment) derived from a secret that is only known to the user. The upcoming problem is that a user is able and motivated to share his pseudonyms and credentials with others if the secret is not of importance. The secret should be of value such that the users are discouraged from sharing. The idea to link the sharing of a credential to

the sharing of a valuable secret key from outside the system is discussed by Lysyanskaya et al. [35] and Dwork et al. [28].

Anonymity revocation Global anonymity revocation is a mechanism for discovering the identity of a user. This is a desirable property to identify users involved in illegal transactions. We speak of local anonymity revocation if a user's pseudonym with an credential issuing organization is revealed.

Attribute encoding The ability to encode attributes into credentials. One example is to encode an expiration date.

Notice, for attribute encoding in the CS model we will present and utilize a signature scheme for blocks of messages. We will describe it in more detail in Chapter 6. Lysyanskaya [34] describes in more detail how to build in additional properties given a basic credential system.

3.2 A Secure Cryptographic System regarding an Ideal-World Specification

The definition of anonymous credential systems by Lysyanskaya [34] is inspired by the work on an universally composable framework from Canetti [19]. The anonymous credential system that we design in this thesis will rely on the framework that is established in this section. In its core the framework is a definition of a basic anonymous credential system proven to be secure in [34]. Secure means that the basic system conforms to the ideal-world specification. The ideal-world specification is originally presented in the same work. The system is called basic since it provides the basic properties of an anonymous credential system as described above.

Definition 3.1 (Ideal-World Specification (IS) [34])**.** An ideal functionality consists of honest ideal players, the adversary $\mathcal{A}$, the environment and a trusted party T. All transactions in this ideal model are carried out by the trusted party T. A transaction is a process that is executed by two or more parties. Those transactions are single round or multiple round communications. Let us now give the specification of the overall system and the parts involved. The security parameter 1^η is given as input to all parties. The environment E controlling the transactions proceeds in periods of time. Transactions are activated by E. Each period starts with E activating one (honest) player $P \in IS$. E instructs the honest player P which transaction it has to initiate. P then contacts the trusted party T and tells it what transaction it intends to carry out and with whom. The period progresses by T executing the trusted code for this transaction. This may involve sending messages to other players and receiving messages from them. T may send or receive messages from an adversary during the execution. The period ends with P telling E the outcome of the transaction. The adversary $\mathcal{A}$ is also controlled by

the environment E. Therefore, $\mathcal{A}$ forwards all received messages to E. Further, E tells $\mathcal{A}$ which messages to send. Besides that $\mathcal{A}$ acts as any other player and is able to send and receive messages to and from T.

Definition 3.2 (Cryptographic System (CS))**.** A cryptographic system is a set of honest cryptographic players, the adversary $\mathcal{A}$, and the environment E. What is missing is the trusted party. The communications are carried out over a network. It is assumed that the network provides secure channels for communication. The CS is initialized by a so called *Init* algorithm. It initializes the necessary public parameters. *Init* is carried out by an honest trusted party which makes the values publicly available. As in the IS the environment proceeds in periods of time. A period consists of the following actions. First E tells one honest player $P \in CS$ which transaction to initiate. Subsequently P contacts the parties it needs to interact with. P then carries out the corresponding cryptographic transaction (protocol) with the contacted parties. Next, P tells E the outcome of the transaction (protocol) and the period ends. Adversary $\mathcal{A}$ is controlled by E and is able to send and receive messages to and from any other player. $\mathcal{A}$ also forwards any received messages to E and E tells $\mathcal{A}$ which message to send and to whom.

The following security definition is essential for the framework established in this chapter. It is originally presented in [34]. It defines what it means for a cryptographic system to be secure in respect to an ideal specification.

Definition 3.3. A cryptographic system CS with initialization algorithm *Init* is secure in the sense of an ideal specification IS in a black-box fashion, if there exists simulator S such that for all pairs of probabilistic polynomial-time families of interactive Turing machines $\{E_\eta, \mathcal{A}_\eta\}$ there exists a negligible function $\nu(\eta)$ such that,

$$\left| \Pr\left[PK \leftarrow Init(1^\eta);\ \left(CS(1^\eta, PK) \longleftrightarrow \mathcal{A}_\eta \longleftrightarrow E_\eta\right) \longrightarrow b\colon b = 1\right] - \right.$$

$$\left. \Pr\left[\left(IS(1^\eta) \longleftrightarrow T(1^\eta) \longleftrightarrow S(\mathcal{A}_\eta) \xrightarrow{\blacksquare} E_\eta\right) \longrightarrow b\colon b = 1\right] \right| = \nu(\eta)$$

The simulator S is given black-box access to the environment. Notice that the arrows stand for communication. Further, notice that in the systems above the environment has to schedule a transaction for a given period. It cannot interleave messages of different transactions or have transactions conducted concurrently.

Next, we will define an ideal basic anonymous credential system in the IS model based on the Section 3.2.1 of [34]. We have adapted the notation and emphasize which parties are involved. After that we will introduce a corresponding system in the CS model.

3.2.1 An Ideal Basic Credential System Functionality

A basic credential system consists of users, organizations and verifiers. Verifiers are players that only verify existing credentials of the users. They can not grant credentials. Users receive credentials and show them to get access. The system is not static. New users can join the set of users over time. Organizations are players that grant and verify credentials of the users. For simplicity reasons we assume that every organization grants a unique type of credential. We call organizations also verifiers if their role in a transaction is to verify a credential. The following credential system fulfills the basic properties unforgeability of credentials, unlinkability of credential showings, consistency of credentials and anonymity of users. The following ideal credential system and corresponding transactions are originally presented in [13] and a variation of it can be found in [34].

Definition 3.4. The ideal basic anonymous credential system consists of the following transactions between users, organizations and verifiers.

FormNym(U, O) A transaction between a user U and an organization O (or verifier). Every time U contacts T it sends a login name L_U known to T and corresponding authenticating key K_U. If this login name is not yet established, U send T the login name L_U and obtains in return the key K_U. U tells T to establish a pseudonym N_O for U with organization O. T first verifies (L_U, K_U) and checks if K_U is the authenticating key for L_U. Then T tells organization O that some user wants to establish a pseudonym N_O. Eventually O either accepts or rejects. If O accepts, it generates a random k-bit tag t and stores it together with N_O. After that T forwards to U the answer (reject or accept) of O.

GrantCred(U, N_O, O) A transaction between a user U and an organization O. U contacts T with the login name L_U, authenticating key K_U, pseudonym N_O and the name of the organization O. T first verifies (L_U, K_U) and checks if K_U is the authenticating key for L_U. If the check fails or if N_O is not U's pseudonym with O, T answers with *Fail*. Otherwise, T proceeds by contacting O. If O accepts, then T tells U that a credential with O on pseudonym N_O has been granted. Otherwise, T tells U *Reject*.

VerifyCred(U, V, N_O, O) A transaction between a user U and a verifier V. U contacts T with the login name L_U, authenticating key K_U, pseudonym N_O, the name of the verifier V and the name of the credential-granting organization O. T first verifies (L_U, K_U) and checks if K_U is the authenticating key for L_U. If the check fails or if N_O is not U's pseudonym with O, T answers with *Fail*. Otherwise, if organization O has granted N_O a credential, then T tells V that the user U has a credential from O.

VerifyCredOnNym(U, V, N_V, N_O, O) A transaction between a user U and a verifier V. U contacts T with the login name L_U, authenticating key K_U,

pseudonym N_O and N_V, the name of the verifier V and the name of the credential-granting organization O. T verifies (L_U, K_U) validity and checks that K_U is the authenticating key for L_U. Further T checks that N_V is the pseudonym established between U and V, N_O is U's pseudonym with O and a credential has been granted by O to N_O. If all checks hold, then T tells V that U with pseudonym N_V has a credential from O.

Notice that to show a party a single credential the *VerifyCred*(U, V, N_O, O) transaction is more efficient, than first running *FormNym*(U, V) to establish N_V (U's pseudonym with V) and then running *VerifyCredOnNym*(U, V, N_V, N_O, O). Further, notice that any organization can also act as an verifier in the transactions. But only organizations can grant a credential using the *GrantCred*(U, N_O, O) transaction.

3.2.2 Designing a Secure Basic Anonymous Credential System based on Signatures and Commitments

The following general definition of a basic anonymous credential system in the CS model is originally presented in [34, Section 3.2.2]. We adapted the notation and our definition is more specific about the used keys in each protocol. The anonymous credential system is called basic, because it provides the basic properties unforgeability of credentials, unlinkability of credential showings, consistency of credentials and anonymity of users. Loosely speaking this is true, since it is the basic system of Definition 3.4 realized in the CS model.

Definition 3.5. The basic anonymous credential system consists of the following transactions between users, organizations and verifiers.

Init Outputs the public parameter 1^η and a public key $pk_{\mathcal{C}}$ of a commitment scheme.

User Initialization The user U chooses a secret M_U.

Organization Initialization Organization O runs $\mathrm{Gen}(1^\eta)$ and gets a signature key pair (sk_O, pk_O). It publishes pk_O as his public key. Notice that the public key pk_O can also include a public key of a commitment scheme. Users will then form pseudonyms with the organization O using the included commitment public key.

FormNym(U, O) User U forms a pseudonym to his secret M_U by forming a commitment to his secret. Let $N_O = \mathrm{Commit}(pk_{\mathcal{C}}, M_U, s)$ for random s. U send N_O to organization O. Further, U proves the knowledge of the committed value. If no failure occurs, then U and O store both N_O as U's pseudonym with O. The organization O also stores a random identifying tag t for the pseudonym.

GrantCred(U, N_O, O) User U and organization O run a protocol for signing a committed value (Section 3.3) on common input $(pk_O, pk_{\mathcal{C}}, N_O)$. Where the user's private input is M_U and the organizations private input is sk_O. Notice that the public key $pk_{\mathcal{C}}$ of the commitment scheme can also be part of the organization public key pk_O. The user stores his output σ of the protocol. Where σ is a valid signature on N_O under sk_O and called credential.

VerifyCred(U, V, N_O, O) User U who has previously participated in the transaction *GrantCred*(U, N_O, O) and obtained the credential σ. Where $N_O = \text{Commit}(pk_{\mathcal{C}}, M_U, s)$. U forms a commitment $N_V = \text{Commit}(pk_{\mathcal{C}}, M_U, r)$ to his secret M_U where r is a random element. User U and verifier V run the protocol for proving knowledge of a signature on a committed value. Where the common input to U and V is $(pk_O, pk_{\mathcal{C}}, N_V)$ and the private input for U is (M_U, r, σ).

VerifyCredOnNym(U, V, N_V, N_O, O) User U who has previously participated in *GrantCred*(U, N_O, O) transaction and obtained the credential σ. Where $N_O = \text{Commit}(pk_{\mathcal{C}}, M_U, s)$. Further, U has established the pseudonym $N_V = \text{Commit}(pk_{\mathcal{C}}, M_U, r)$ with V. User U and verifier V run the protocol for proving knowledge of a signature on a committed value. Where the common input to U and V is $(pk_O, pk_{\mathcal{C}}, N_V)$ and the private input for U is (M_U, r, σ).

Theorem 3.6. *The basic anonymous credential system of Definition 3.5 is secure in terms of Definition 3.3.*

The proof is presented in [34] by Lysyanskaya and in a more specific form in [13] by Camenisch and Lysyanskaya. In short, to prove the security one has to define a simulator S that translates the CS system environment (real environment) and adversary into an ideal environment and vice versa.

3.3 Signing a Committed Value

In this section, we will define the formal outline of a two-party protocol to obtain a signature on a committed value. We will use this protocol to issue credentials on pseudonyms. Hence, it is a building block for the design of an anonymous credential system. After that, the security of the so called two-party protocol (computation) between a user and a signer will be defined. The result of the protocol is a signature on a committed value of the user's choice. The signer will not learn any information about the users's value. The following definition is originally presented in [34].

Definition 3.7 (Two-party protocol for signing a committed value)**.** Let $\Pi = (\text{Gen}, \text{Sign}, \text{Verify})$ be a secure signature scheme and $\mathcal{C} = (\text{Gen}_{\mathcal{C}}, Commit)$ be a non-interactive commitment scheme. Further, let X_η and R_η be finite sets,

$(pk_O, sk_O) \leftarrow \mathrm{Gen}(1^\eta)$ and $pk_\mathcal{C} \leftarrow \mathrm{Gen}_\mathcal{C}(1^\eta)$. The functional specification of the protocol $(O \leftrightarrow U)$ for signing a committed value between a user U and a signer O is as follows:

Common Input: The signature public key pk_O, commitment public key $pk_\mathcal{C}$ and a commitment C.

User's private Input: The private value $x \in X_\eta$ that needs to be signed and private value $r \in R_\eta$ such that $C = Commit(pk_\mathcal{C}, x, r)$.

Signer's private Input: Secret key sk_O that corresponds to pk_O.

User's output: The signature σ such that $\mathrm{Verify}(pk_O, x, \sigma) = 1$.

The security definition of the two-party protocol $(O \leftrightarrow U)$ for signing a committed value follows the definition presented in [34]. From the view of the user U the protocol $(O \leftrightarrow U)$ must reveal (almost) no information about the user's private input x. Regarding the security for the signer, a (malicious) user cannot output a valid signature σ on any value unless σ was issued by the signer. In the following definition we assume that the public keys are known to every party. With $b \leftarrow (\mathcal{A} \leftrightarrow U(\cdot,\cdot))$ we denote the random variable b representing $\mathcal{A}$'s outputs after the interaction with U. Analogous we denote with $(O(\cdot) \leftrightarrow U(\cdot)) \rightarrow b$ the random variable b representing the output of U after the interaction with O.

Definition 3.8 (Secure two-party protocol)**.** Let $(O \leftrightarrow U)$ be a two-party protocol for signing a committed value as in Definition 3.7. $(O \leftrightarrow U)$ is called secure if the following holds.

- **Security for the user:** There exists a negligible function $\nu(\eta)$ such that for all $x, y \in X_\eta$, for all ppt adversaries $\mathcal{A}$, for all $pk_O \in \mathrm{Gen}(1^\eta)$ and all $pk_C \leftarrow \mathrm{Gen}_\mathcal{C}(1^\eta)$,

$$\left| \Pr\left[r \leftarrow R_\eta, b \leftarrow (\mathcal{A} \leftrightarrow U(x,r)) : b = 0\right] - \Pr\left[r \leftarrow R_\eta, b \leftarrow (\mathcal{A} \leftrightarrow U(y,r)) : b = 0\right] \right| \leq \nu(\eta)$$

- **Security for the signer:** Let O denote the interactive Turing-machine for the signer's part of the protocol. There exists an ppt extractor algorithm $\mathcal{E}$ such that the user's view in interaction with $\mathcal{E}$ is the same as in interactions with O. This holds even though $\mathcal{E}$ only has a **single** oracle access to the signing oracle of the underlying secure signature scheme. More formally, for all probabilistic polynomial-time families of Turing machines $\{U_\eta\}$, there

exists a negligible function $\nu(\eta)$ such that

$$\Big| \Pr\left[(pk_O, sk_O) \leftarrow \mathrm{Gen}(1^\eta), (O(sk_O) \leftrightarrow U_\eta(pk_O)) \rightarrow b\colon b = 0\right] -$$

$$\Pr\left[(pk_O, sk_O) \leftarrow \mathrm{Gen}(1^\eta), \left(\mathcal{E}^{1\,\mathrm{Sign}(pk_O, sk_O, \cdot)}(1^\eta) \xrightarrow{\blacksquare} U_\eta(pk_O)\right) \rightarrow b\colon b = 0\right] \Big| = \nu(\eta)$$

More strongly, let $[O(\cdot)]$ denote the sequentially composed signer protocol, This means a protocol that repeats the signer's part of the protocol sequentially with the adversary U until the adversary stops it. Let the sequentially composed extractor $\left[\mathcal{E}^{(\cdot)}(\cdot)\right]$ be defined analogously. We require the following. For all probabilistic polynomial-time families of Turing machines $\{U_\eta\}$, there exists a negligible function $\nu(\eta)$ such that

$$\Big| \Pr\left[(pk_O, sk_O) \leftarrow \mathrm{Gen}(1^\eta), ([O(sk_O)] \leftrightarrow U_\eta(pk_O)) \rightarrow b\colon b = 0\right] -$$

$$\Pr\left[(pk_O, sk_O) \leftarrow \mathrm{Gen}(1^\eta), \left(\left[\mathcal{E}^{1\,\mathrm{Sign}(pk_O, sk_O, \cdot)}(1^\eta)\right] \xrightarrow{\blacksquare} U_\eta(pk_O)\right) \rightarrow b\colon b = 0\right] \Big| = \nu(\eta)$$

Notice, that the sequentially composable requirement is related to the sequential composition of zero-knowledge proofs [30, Section 4.3.4].

As a building block for an anonymous credential system, the above protocol for signing a committed value has to ensure, that no adversarial user can produce a valid signature on his own without running the protocol with the signer. The following lemma and proof is originally presented by Lysyanskaya in [34].

Lemma 3.9. *Let* $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ *be the secure signature scheme used in the protocol for signing a committed value Definition 3.7 and let* $(pk_O, sk_O) \leftarrow \mathrm{Gen}$. *No polynomial-time adversary* $\mathcal{A}$ *can output a signature* σ *on a value* m*, such that* $\mathrm{Verify}(pk_O, m, \sigma) = 1$ *holds, for which it did not run the protocol for signing a committed value with the signer* O.

Proof. From the *security for the signer* of the protocol for signing a committed value we get that the adversary $\mathcal{A}$ cannot distinguish whether he is talking to the extractor $\mathcal{E}$ or to the actual signer O. Hence, $\mathcal{A}$'s probability to output a forgery when talking to O is as when talking to $\mathcal{E}$. If the adversary outputs a forgery when talking to $\mathcal{E}$ then it is also a direct forgery on the signature scheme. Since the signature scheme Π is secure in the sense of *adaptive chosen-message attack*s this only happens with negligible probability. □

Following the work of Camenisch and Lysyanskaya [13, 14, 34, 15] and the definitions presented in this chapter, an anonymous credential system can be designed with,

- a commitment scheme,
- a signature scheme and

- efficient protocols. Where the protocols are as follows. A protocol for,
  - proving knowledge of a committed value,
  - proving equality of two committed values (optional),
  - obtaining a signature on a committed value and
  - proving knowledge of a signature on a committed value.

In the main part of this thesis we will show how to design an anonymous credential system based on the q-SDH Assumption. We have already presented a commitment scheme for blocks of values (Definition 2.10), a protocol for proving knowledge of a committed value (Protocol 2.25) and a protocol for proving the equality of committed values (Protocol 2.30). What we will present in the following chapters are signature schemes secure under the q-SDH Assumption (Chapter 4), a protocol for obtaining a signature on a committed value and a protocol for proving knowledge of a signature on a committed value (Chapter 5). Finally, in Chapter 6 we will present our anonymous credential system based on the q-SDH Assumption in detail.

4 Signature Schemes based on the q-SDH Assumption

In this chapter we will introduce signature schemes secure under the q-SDH Assumption. The signature schemes are an essential part of our anonymous credential system. Credentials in our anonymous credential system will be generated using the signature schemes presented here. Besides that the signature schemes are of interest on its own and can be used in other systems. The idea for the signature schemes is first mentioned by Camenisch and Lysyanskaya in [15]. Similar signature schemes secure under the q-SDH Assumption are presented in [2, 6, 7].

4.1 A Basic Signature Scheme

The basic signature scheme presented in this section provides signatures for a single message. We will extend it to a signature scheme that supports signing blocks of messages. The extended scheme will be used to sign an information-theoretical hidden message. The information-theoretical hidden message will be a commitment of our commitment scheme from Definition 2.10. But first let us concentrate on the basic scheme. The concept of our basic signature scheme is similar the scheme A of Camenisch and Lysyanskaya [15].

4.1.1 Scheme BBS-A

Definition 4.1 (BBS-A)**.** Let $\mathcal{G}$ be a bilinear group generator. The signature scheme $\Pi_A = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ is defined as follows:

Gen(1^η):

1. Get $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$
2. Choose $g_2 \leftarrow \mathbb{G}_2$, $u_0 \leftarrow \mathbb{G}_2$, $u_1 \leftarrow \mathbb{G}_2$ and set $g_1 := \psi(g_2)$
3. Choose $\gamma \leftarrow \mathbb{Z}_p$ and set $w := g_2^\gamma$
4. Set public key $pk := (g_1, g_2, u_0, u_1, w, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$ and secret key $sk := \gamma$

Sign(pk, sk, m):

1. On input public key $pk = (g_1, g_2, u_0, u_1, w, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$, secret key $sk = \gamma$ and message $m \in \mathbb{Z}_p$
2. Choose $x \leftarrow \mathbb{Z}_p$ and $s \leftarrow \mathbb{Z}_p$
3. Set $h_0 := \psi(u_0)$ and $h_1 := \psi(u_1)$
4. $A = (g_1 \cdot h_0^s \cdot h_1^m)^{\frac{1}{x+\gamma}}$
5. $\sigma := (A, x, s)$

Verify(pk, m, σ):

1. On input public key $pk = (g_1, g_2, u_0, u_1, w, L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$, message m and signature $\sigma = (A, x, s)$
2. Set $h_0 := \psi(u_0)$ and $h_1 := \psi(u_1)$
3. Output 1 if $e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m}, g_2) = e(g_1, g_2)$ holds. Otherwise output 0.

Lemma 4.2. *The* BBS-A *signature scheme is correct.*

Proof. Assume $\sigma \leftarrow Sign(sk, m)$, then the following holds. The signature is $\sigma = (A, x, s)$ with $A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m} = g_1$ and $x, s \in \mathbb{Z}_p$. Verify(pk, σ, m) checks that the following holds.

$$\begin{aligned}
& e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m}, g_2) \\
\Leftrightarrow\ & e(A, g_2^x) \cdot e(A, g_2^\gamma) \cdot e(h_0^{-s} \cdot h_1^{-m}, g_2) \\
\Leftrightarrow\ & e(A, g_2^{x+\gamma}) \cdot e(h_0^{-s} \cdot h_1^{-m}, g_2) \\
\Leftrightarrow\ & e(A^{x+\gamma}, g_2) \cdot e(h_0^{-s} \cdot h_1^{-m}, g_2) \\
\Leftrightarrow\ & e(A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m}, g_2) \\
\Leftrightarrow\ & e(g_1, g_2)
\end{aligned}$$

□

Theorem 4.3. *Under the q-SDH Assumption for bilinear group generator $\mathcal{G}$, the* BBS-A *signature scheme is secure against existential forgery under an adaptive chosen-message attack.*

Proof. The theorem is a special case of Theorem 4.10 with $L = 1$. Thus, the security follows directly from the security of the BBS-B signature scheme (Definition 4.4), presented in the next section, with L set to 1. □

We note that this signature scheme is sufficient for the design of a group signature scheme. Whereby the user chooses m and give h_1^m to the group manager. The user proves the knowledge of m and obtains the membership certificate $(A = (g_1 \cdot h_0^s \cdot h_1^m)^{\frac{1}{x+\gamma}},\ x,\ s)$.

4.2 A Signature Scheme for Blocks of Messages

We generalize the BBS-A signature scheme, presented in Section 4.1, to a signature scheme for blocks of messages of length L. We prove the security of the signature scheme based on the q-SDH Assumption. The proof is also applicable for the BBS-A scheme with L set to 1. The signature scheme for a block of message is of interest for our anonymous credential system, since it supports attribute encoding, see 3.1.2. This allows us to grant credentials on commitments, which not only include a user's secrets, but also include attributes such as expiration dates. Further it can be used to design a protocol for signing an information-theoretical hidden message with $L \geq 2$. In this regard it can be compared to the schemes B and C of Camenisch and Lysyanskaya [15]. A similar signature scheme is presented by Au et al. [2] and is also based on the idea mentioned by Camenisch and Lysyanskaya in [15]. As mentioned before, the following signature scheme is essential for our anonymous credential system to grant credentials on pseudonyms. Thus, it will be used in our protocol for signing a committed value. Note that we assume that the block size L is fixed. Therefore the public key is generated for a specific value L.

4.2.1 Scheme BBS-B

Definition 4.4 (BBS-B)**.** Let $\mathcal{G}$ be a bilinear group generator. The signature scheme $\Pi_B = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ is defined as follows:

Gen(1^η):

1. Get $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$
2. Choose $g_2 \leftarrow \mathbb{G}_2$, $u_0, \ldots, u_L \leftarrow \mathbb{G}_2$ and set $g_1 := \psi(g_2)$
3. Choose $\gamma \leftarrow \mathbb{Z}_p$ and set $w := g_2^\gamma$
4. Set public key $pk := (g_1, g_2, u_0, \ldots, u_L, w, L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$ and secret key $sk := \gamma$

Sign(pk, sk, M):

1. On input public key $pk = (g_1, g_2, u_0, \ldots, u_L, w, L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$, secret key $sk = \gamma$ and message block $M = (m_1, \ldots, m_L) \in \mathbb{Z}_p^L$
2. Choose $x \leftarrow \mathbb{Z}_p$ and $s \leftarrow \mathbb{Z}_p$
3. Set $h_i := \psi(u_i)$ for $i = 0, \ldots, L$
4. Set $A := (g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L})^{\frac{1}{x+\gamma}}$
5. Output $\sigma := (A, x, s)$

Verify(pk, M, σ):

1. On input public key $pk = (g_1, g_2, u_0, u_1, \ldots, u_L, w, L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$, message block $M = (m_1, \ldots, m_L)$ and signature $\sigma = (A, x, s)$
2. Set $h_i := \psi(u_i)$ for $i = 0, \ldots, L$
3. Output 1 if $e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) = e(g_1, g_2)$ holds. Otherwise output 0.

Lemma 4.5. BBS-B *signature scheme is a correct.*

Proof. Assume $\sigma \leftarrow \text{Sign}(pk, sk, M)$ where $M = (m_1, \ldots, m_L)$, then $\sigma = (A, x, s)$ with $A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L} = g_1$ and $x, s, m_i \in \mathbb{Z}_p$ for $i = 1, \ldots, L$. Then $\text{Verify}(pk, m, \sigma)$ checks if the following holds:

$$\begin{aligned}
& e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(A, g_2^x) \cdot e(A, g_2^\gamma) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(A, g_2^{x+\gamma}) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(g_1, g_2)
\end{aligned}$$

Hence, $\text{Verify}\,(pk, m, \text{Sign}(pk, sk, M))$ outputs 1 for every message block $M = (m_1, \ldots, m_L) \in \mathbb{Z}_p^L$ and every key pair $(pk, sk) \leftarrow \text{Gen}(1^\eta)$. □

Next, we will prove that the BBS-B signature scheme is secure. Let us first give an informal outline. Assume $\mathcal{F}$ is a forger for signature scheme BBS-B with success probability $\varepsilon_\mathcal{F}$ in $\text{Sig-forge}_{\mathcal{F},\Pi_B}(\eta)$ and running time $t_\mathcal{F}$ that makes q signature queries. Using $\mathcal{F}$ we can then define an adversary $\mathcal{A}$, which solves the q-SDH problem for bilinear group generator $\mathcal{G}$ (respectively in $(\mathbb{G}_1,\mathbb{G}_2)$). We will first distinguish three types of outputs of $\mathcal{F}$ (Type 1, Type 2, Type 3). We will show that any of these will lead to $\mathcal{A}$ solving the q-SDH problem for bilinear group generator $\mathcal{G}$. Indeed we will show that Type 1 and Type 2 forgery allow $\mathcal{A}$ to solve a given q-SDH problem instance with a variant of the technique presented in Lemma 3.2 of [6] by Boneh and Boyen. Type 3 forgery leads to solving the discrete logarithm problem in $\mathbb{G}_1$. Which is assumed to be hard for the groups that we consider as output of $\mathcal{G}$.

More formally, suppose $\mathcal{F}$'s q signature queries for message blocks $M_1, \ldots M_q$ are answered with $\sigma_i = (A_i, x_i, s_i)$ for $i = 1, \ldots, q$. Further, suppose that the output of $\mathcal{F}$ is a forgery $\sigma' = (A', x', s')$ on message block M', then there are three types of outputs.

Type 1: $\mathcal{F}$ outputs a forgery $\sigma' = (A', x', s')$ on message block M' where $x' \notin \{x_1, \ldots, x_q\}$.

Type 2: $\mathcal{F}$ outputs a forgery $\sigma' = (A', x', s')$ on message block M' where $x' \in \{x_1, \ldots, x_q\}$ ($x' = x_k$ for some $k \in \{1, \ldots, q\}$) and $A' \neq A_k$.

Type 3: $\mathcal{F}$ outputs a forgery $\sigma' = (A', x', s')$ on message block M' where $x' \in \{x_1, \ldots, x_q\}$ ($x' = x_k$ for some $k \in \{1, \ldots, q\}$) and $A' = A_k$.

The same technique, for different output types, is used in [34]. Let $\mathcal{F}_1$ (respectively $\mathcal{F}_2$ or $\mathcal{F}_3$) denote the forger who runs $\mathcal{F}$, but only outputs the forgery if it is a Type 1 (respectively Type 2 or Type 3) output.

Lemma 4.6. *If the success probability of forger $\mathcal{F}$ is $\varepsilon_{\mathcal{F}}$ then one of $\mathcal{F}_1$, $\mathcal{F}_2$ or $\mathcal{F}_3$ succeeds with probability at least $\varepsilon_{\mathcal{F}}/3$.*

Proof. The above lemma follows from the union bound. □

Before we go into more detail, let us give an informal description of the structure of the proof. The adversary that we will describe will first generate keys for the signature scheme to initialize the forger. For the key generation we will use the technique presented by Boneh and Boyen in [6]. After that we are are able to generate $q - 1$ valid q-SDH pairs. We will answer $q - 1$ signature queries of the forger by using these pairs. One uniformly at random chosen signature query will be answered using specially chosen values. We will go into more detail about these values in the actual proof. But let us state here, that these values will allow us to solve the given q-SDH problem in a case, where otherwise we could not bound the success probability of the adversary. This case will be considered in Lemma 4.8 for forger $\mathcal{F}_2$.

Next, we will proceed with proving that the success probability of each of the forgers $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ are negligible under the corresponding assumptions.

Lemma 4.7. *Under the q-SDH Assumption, the success probability of forger $\mathcal{F}_1$ in the Sig-forge$_{\mathcal{F}_1,\Pi_B}(\eta)$ game is negligible.*

Proof. First assume that the success probability of $\mathcal{F}_1$ in Sig-forge$_{\mathcal{F}_1,\Pi_B}(\eta)$ is ε_1 and its running time is t_1. Let us show that this contradicts the q-SDH Assumption. We construct an algorithm $\mathcal{A}$ that given a q-SDH instance $(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, \ldots, g_2^{\gamma^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, corresponding $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$ and access to $\mathcal{F}_1$ will output a valid SDH pair $(A, x') \in \mathbb{G}_1 \times \mathbb{Z}_p$ such that $A = g_1^{\frac{1}{x'+\gamma}}$.

Key Generation: Adversary $\mathcal{A}$ obtains generators g_1',g_2',$w = g_2'^{\gamma}$ and SDH pairs with the technique presented in Lemma 3.2 of [6] as follows. It first defines $B_i := g_2^{\gamma^i}$ for $i = 0, \ldots, q$. Let f be the polynomial defined as $f(\Gamma) = \prod_{i=1}^{q-1}(\Gamma + x_i)$ and expand it. Then write it as $f(\gamma) = \sum_{i=0}^{q-1} \alpha_i \Gamma^i$ where $\alpha_i \in \mathbb{Z}_p$ for $i = 0, \ldots, q-1$. Notice that $\mathcal{A}$ can efficiently calculate the

coefficients $\alpha_0, \ldots, \alpha_{q-1}$ from f.
Next, $\mathcal{A}$ computes

$$g_2' = \prod_{i=0}^{q-1} B_i^{\alpha_i} = g_2^{\sum_{i=0}^{q-1} \alpha_i \gamma^i} = g_2^{f(\gamma)}, \qquad w' = \prod_{i=i}^{q} B_i^{\alpha_i} = g_2^{\gamma f(\gamma)} = g_2'^{\gamma}$$

and sets $g_1' = \psi(g_2')$. It now computes SDH pairs (D_i, x_i) for $i = 1, \ldots, q-1$. Therefore, let $f_i(\Gamma) = f(\Gamma)/(x_i + \Gamma) = \prod_{j=1, j\neq i}^{q-1}(x_j + \Gamma)$ be the polynomial f_i and expand it to $f_i(\Gamma) = \sum_{j=0.}^{q-2} \beta_j \Gamma^j$. The values $\beta_j \in \mathbb{Z}_p$ for $j = 0, \ldots, q-2$ can be efficiently calculated from x_l where $l \neq i$ and $l = 1, \ldots, q-1$. The elements D_i for $i = 1, \ldots, q-1$ of the SDH pairs are set to $D_i := (g_1')^{\frac{1}{x_i+\gamma}} = \psi(E_i)$, where $E_i = \prod_{j=0}^{q-2} B_j^{\beta_j} = g_2^{\sum_{j=0.}^{q-2} \beta_j \gamma^j} = g_2^{f_i(\gamma)} = g_2^{f(\gamma)/(x_i+\gamma)} = (g_2')^{\frac{1}{x_i+\gamma}}$. Notice that we may assume that $f(\Gamma) \neq 0$. Otherwise $-x_i = \gamma$ for some i. This means $\mathcal{A}$ successfully obtained the secret key γ. Since $x_i \leftarrow \mathbb{Z}_p$, this only happens with negligible probability. The computed pairs (D_i, x_i) are valid SDH pairs such that $e(D_i, w g_2'^{x_i}) = e(g_1', g_2')$. $\mathcal{A}$ has now generated g_1', g_2', $w = g_2'^{\gamma}$ and the SDH pairs (D_i, x_i) for $i = 1, \ldots, q-1$.
Next, $\mathcal{A}$ computes u_0, which will help solving the given q-SDH problem. Therefore, it chooses $x^*, v^*, r^* \leftarrow \mathbb{Z}_p$, computes

$$u_0 = \left((w g_2'^{x^*})^{r^*} / g_2'\right)^{1/v^*} = g_2'^{\frac{(x^*+\gamma)r^*-1}{v^*}}$$

and sets $u_i = u_0^{\mu_i}$ where $\mu_i \leftarrow \mathbb{Z}_p$ for $i = 1, \ldots, L$. $\mathcal{A}$ initializes forger $\mathcal{F}_1$ with the public key $pk := (g_1', g_2', u_0, u_1, \ldots, u_L, w,\ L,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ p,\ \psi,\ e)$. Notice, in the following we will use $h_0 = g_1'^{\frac{(x^*+\gamma)r^*-1}{v^*}} = \psi(u_0)$ and $h_i = g_1'^{\left(\frac{(x^*+\gamma)r^*-1}{v^*}\right)^{\mu_i}} = h_0^{\mu_i} = \psi(u_i)$ for $i = 1, \ldots, L$. Further, notice that the values x^*, v^* and r^* will also be used to answer one of the q signature queries.

Signing: Let $M_i = m_{i,1}, \ldots, m_{i,L}$ be the message block of the i-th query. $\mathcal{A}$ sets $t_i = m_{i,1}\mu_1 + \ldots + m_{i,L}\mu_L$ after getting a signature query for message block M_i. Adversary $\mathcal{A}$ has to answer q signature queries but has only $q-1$ valid SDH pairs (D_i, x_i). Therefore $\mathcal{A}$ determines up front uniformly at random one of the q signature queries of $\mathcal{F}_1$. We denote this query with i^*. Query i^* is answered by $\mathcal{A}$ through computing $A^* = g_1'^{r^*}$ and choosing $s^* \in \mathbb{Z}_p$ such that $v^* = s^* + t^* \mod p$, where $t^* = m_1^*\mu_1 + \ldots + m_L^*\mu_L \mod p$. Then it returns $\sigma^* := (A^*, x^*, s^*)$ as the signature. Let us show that this is a valid signature, i.e. $\text{Verify}(pk, M^*, \sigma^*) = 1$ where $M^* = m_1^*, \ldots, m_L^*$. For this we

show that $e(A^*, g_2')^{x^*} \cdot e(A^*, w') \cdot e(h_0^{-s^*} \cdot h_1^{-m_1^*} \cdot \ldots \cdot h_L^{-m_L^*}, g_2') = e(g_1', g_2')$ holds.

$$
\begin{aligned}
& e(A^*, g_2')^{x^*} \cdot e(A^*, w') \cdot e(h_0^{-s^*} \cdot h_1^{-m_1^*} \cdot \ldots \cdot h_L^{-m_L^*}, g_2') \\
=& e(A^*, g_2'^{x^*}) \cdot e(A^*, g_2'^{\gamma}) \cdot e(g_1'^{\left(\frac{(x^*+\gamma)r^*-1}{v^*}\right)^{-s^*}} \cdot g_1'^{\left(\frac{(x^*+\gamma)r^*-1}{v^*}\right)^{-t^*}}, g_2') \\
=& e(g_1'^{r^*}, g_2'^{x^*+\gamma}) \cdot e(g_1'^{\left(\frac{(x^*+\gamma)r^*-1}{v^*}\right)^{-v^*}}, g_2') \\
=& e(g_1'^{r^*}, g_2'^{x^*+\gamma}) \cdot e(g_1'^{\left(\frac{-v^*(x^*+\gamma)r^*+v^*}{v^*}\right)}, g_2') \\
=& e(g_1'^{r^*(x^*+\gamma)}, g_2') \cdot e(g_1'^{-r^*(x^*+\gamma)+1}, g_2') \\
=& e(g_1'^{r^*(x^*+\gamma)} \cdot g_1'^{-r^*(x^*+\gamma)+1}, g_2') \\
=& e(g_1', g_2')
\end{aligned}
$$

Observe that the element A is a random element in $\mathbb{G}_1$ and x^*, s^* are random elements in $\mathbb{Z}_p$ as in the real signature scheme (resp. game).
For the other $q-1$ queries let i denote the current query. $\mathcal{A}$ chooses $s_i \leftarrow \mathbb{Z}_p$ and computes $v_i = s_i + t_i \mod p$, where $t_i = m_{i,1}\mu_1 + \ldots + m_{i,L}\mu_L \mod p$. Notice that $h_0^{v_i} = h_0^{s_i} \cdot h_1^{m_{i,1}} \cdot \ldots \cdot h_L^{m_{i,L}}$, $D_i = (g_1')^{\frac{1}{x_i+\gamma}}$ and $h_j = h_0^{\mu_j}$ for $j = 1, \ldots, L$. $\mathcal{A}$ computes the signature $\sigma_i := (A_i, x_i, s_i)$ on $M_i = m_{i,1}, \ldots, m_{i,L}$ as follows:

$$
\begin{aligned}
A_i &= (g_1' \cdot h_0^{s_i} \cdot h_1^{m_{i,1}} \cdot \ldots \cdot h_L^{m_{i,L}})^{\frac{1}{x_i+\gamma}} \\
&= (g_1' \cdot h_0^{v_i})^{\frac{1}{x_i+\gamma}} \\
&= g_1'^{\frac{1}{x_i+\gamma}} \cdot h_0^{\frac{v_i}{x_i+\gamma}} \\
&= D_i \cdot g_1'^{\frac{(x^*+\gamma)r^*-1}{v^*} \cdot \frac{v_i}{x_i+\gamma}} \\
&= D_i \cdot g_1'^{\frac{v_i(x^*+\gamma)r^*-v_i}{v^*(x_i+\gamma)}} \\
&= D_i \cdot g_1'^{\frac{-v_i}{v^*(x_i+\gamma)}} \cdot g_1'^{\frac{v_i(x^*+\gamma)r^*}{v^*(x_i+\gamma)}} \\
&= D_i \cdot D_i^{-\frac{v_i}{v^*}} \cdot g_1'^{\frac{v_i(x^*+\gamma)r^*}{v^*(x_i+\gamma)}} \\
&= D_i^{1-\frac{v_i}{v^*}} \cdot g_1'^{\frac{v_i r^* x_i - v_i r^* x_i + v_i r^* x^* + v_i r^* \gamma}{v^*(x_i+\gamma)}} \\
&= D_i^{1-\frac{v_i}{v^*}} \cdot g_1'^{\frac{v_i r^*(x_i+\gamma) - v_i r^*(x_i - x^*)}{v^*(x_i+\gamma)}} \\
&= D_i^{1-\frac{v_i}{v^*}} \cdot g_1'^{\frac{v_i r^*}{v^*} - \frac{v_i r^*(x_i-x^*)}{v^*(x_i+\gamma)}} \\
&= D_i^{1-\frac{v_i}{v^*}} \cdot g_1'^{\frac{v_i r^*}{v^*}} \cdot g_1'^{-\frac{v_i r^*(x_i-x^*)}{v^*(x_i+\gamma)}} \\
&= D_i^{1-\frac{v_i}{v^*} - \frac{v_i r^*(x_i-x^*)}{v^*}} \cdot g_1'^{\frac{v_i r^*}{v^*}}
\end{aligned}
$$

Observe that the signature $\sigma_i := (A_i, x_i, s_i)$ on $M_i = m_{i,1}, \ldots, m_{i,L}$ is valid, i.e. $\text{Verify}(pk, M_i, \sigma_i) = 1$. Further, the elements are correctly distributed. This is true since, $\mathcal{A}$ chooses s_i and x_i uniformly at random from $\mathbb{Z}_p$ as in the real game.

Processing the Forgery Eventually $\mathcal{F}_1$ will output a forgery $\sigma' = (A', x', s')$ on message block M'. Since $\mathcal{F}_1$ only outputs Type 1 forgeries it holds that $x' \notin \{x_1, \ldots, x_q\}$. This means we can get a new SDH pair from $\mathcal{F}_1$'s forgery as follows. Let $v' = s' + m'_1\mu_1 + \ldots + m'_L\mu_L \mod p$, then the following holds.

$$A'^{x'+\gamma} = g'_1 \cdot h_0^{v'}$$
$$A'^{x'+\gamma} = g'_1 \cdot g_1'^{\frac{r^* v'(x^*+\gamma)-v'}{v^*}}$$

Which is equivalent to,

$$\begin{aligned}
A' &= g_1'^{\frac{1}{x'+\gamma}} \cdot g_1'^{\frac{r^* v'(x^*+\gamma)-v'}{v^*(x'+\gamma)}} \\
&= g_1'^{\frac{1}{x'+\gamma}} \cdot g_1'^{-\frac{v'}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v' x^* + r^* v'\gamma}{v^*(x'+\gamma)}} \\
&= g_1'^{\frac{v^*}{v^*(x'+\gamma)}} \cdot g_1'^{-\frac{v'}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v' x' - r^* v' x' + r^* v' x^* + r^* v'\gamma}{v^*(x'+\gamma)}} \\
&= g_1'^{\frac{v^*-v'}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v'(x'+\gamma) - r^* v'(x'-x^*)}{v^*(x'+\gamma)}} \\
&= g_1'^{\frac{v^*-v'}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v'}{v^*}} \cdot g_1'^{-\frac{r^* v'(x'-x^*)}{v^*(x'+\gamma)}} \\
&= g_1'^{\frac{v^*-v'-r^* v'(x'-x^*)}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v'}{v^*}} && \Leftrightarrow \\
g_1'^{\frac{1}{v^*(x'+\gamma)}} &= \left(A' g_1'^{-\frac{r^* v'}{v^*}}\right)^{\frac{1}{v^*-v'-r^* v'(x'-x^*)}} && \Leftrightarrow \\
g_1'^{\frac{1}{(x'+\gamma)}} &= \left(A' g_1'^{-\frac{r^* v'}{v^*}}\right)^{\frac{v^*}{v^*-v'-r^* v'(x'-x^*)}}
\end{aligned}$$

Adversary $\mathcal{A}$ can now output a solution to the given q-SDH instance using the new SDH pair (D', x') where $D' := g_1'^{\frac{1}{(x'+\gamma)}}$. The technique that we will use in the following is originally presented in Lemma 3.2 of [6]. Notice that the following holds for (D', x'), $e(D', w g_2'^{x'}) = e(D', g_2'^{x'+\gamma}) = e(g_1'^{\frac{x'+\gamma}{x'+\gamma}}, g'_2) = e(g'_1, g'_2)$. From $g'_1 = g_1^{f(\gamma)}$ we get that $D' = g_1'^{\frac{1}{(x'+\gamma)}} = g_1^{\frac{f(\gamma)}{(x'+\gamma)}}$. We rewrite the polynomial $f(\Gamma)$ as $f(\Gamma) = \chi(\Gamma)(x' + \Gamma) + c$ for some polynomial

$\chi(\Gamma) = \sum_{i=0}^{q-2} \chi_i \Gamma^i$ and $c \in \mathbb{Z}_p$. We rewrite $f(\Gamma)/(x' + \Gamma)$ as

$$\frac{f(\Gamma)}{(x' + \Gamma)} = \frac{c}{x' + \Gamma} + \sum_{i=0}^{q-2} \chi_i \Gamma^i$$
$$\text{and } D' = g_1^{\frac{c}{x'+\gamma} + \sum_{i=0}^{q-2} \chi_i \gamma^i}$$

Then $\mathcal{A}$ computes A as follows,

$$\begin{aligned} A &= \left(D' \cdot \prod_{i=0}^{q-2} \psi(B_i)^{-\chi_i} \right)^{\frac{1}{c}} \\ &= \left(g_1^{\frac{c}{x'+\gamma} + \sum_{i=0}^{q-2} \chi_i \gamma^i} \cdot \prod_{i=0}^{q-2} g_1^{-\chi_i \gamma^i} \right)^{\frac{1}{c}} \\ &= \left(g_1^{\frac{c}{x'+\gamma} + \sum_{i=0}^{q-2} \chi_i \gamma^i} \cdot g_1^{-\sum_{i=0}^{q-2} \chi_i \gamma^i} \right)^{\frac{1}{c}} \\ &= g_1^{\frac{1}{x'+\gamma}} \end{aligned}$$

Finally $\mathcal{A}$ returns (A, x') as his solution to the given q-SDH instance.

Success probability $\mathcal{A}$ perfectly simulates the Sig-forge$_{\mathcal{F}_1,\Pi_B}(\eta)$ game. Further, $\mathcal{A}$ succeeds in breaking the q-SDH Assumption for $\mathcal{G}$, whenever $\mathcal{F}_1$ outputs a forgery. Hence, the success probability of $\mathcal{A}$ is ε_1. Its running time is $t_1 + q\mathcal{O}(1)$. Since, the q-SDH Assumption holds for $\mathcal{G}$, $\mathcal{A}$'s success probability is negligible.

□

Lemma 4.8. *Under the q-SDH Assumption, the success probability of forger $\mathcal{F}_2$ in the Sig-forge$_{\mathcal{F}_2,\Pi_B}(\eta)$ game is negligible.*

Proof. Let $\mathcal{F}_2$ be the forger with success probability ε_2 in the Sig-forge$_{\mathcal{F}_2,\Pi_B}(\eta)$ game and running time t_2. We show that this contradicts the q-SDH Assumption for bilinear group generator $\mathcal{G}$. We construct an algorithm $\mathcal{A}$ that given a q-SDH instance $(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, \ldots, g_2^{\gamma^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, corresponding ($\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, p, ψ, e)$\leftarrow \mathcal{G}(1^\eta)$ and access to $\mathcal{F}_2$ will output a valid SDH pair (A, x') such that $A = g_1^{\frac{1}{x'+\gamma}}$, $A \in \mathbb{G}_1$ and $x' \in \mathbb{Z}_p$.

Key Generation: $\mathcal{A}$ works as in Lemma 4.7

Signing: $\mathcal{A}$ answers the signature queries of $\mathcal{F}_2$ as in Lemma 4.7.

Processing the Forgery Eventually $\mathcal{F}_2$ will output a forgery $\sigma' = (A', x', s')$ on message block M'. Since $\mathcal{F}_2$ only outputs Type 2 forgeries it holds that

$x' \in \{x_1, \ldots, x_q\}$ ($x' = x_k$ for some $k \in \{1, \ldots, q\}$) and $A' \neq A_k$. Notice that query i^* was a special case and answered by $\mathcal{A}$ with $\sigma^* = (A^*, x^*, s^*)$. If $x' \in \{x_1, \ldots, x_q\}$ and $x' \neq x^*$ we have that $x' = x_i$ and $A' \neq A_i$. Where x_i and A_i were part of an answer to the i-th signature query for $i \neq i^*$. Especially x_i is already a part of a known SDH pair and does not help $\mathcal{A}$ to solve the given q-SDH problem. Then $\mathcal{A}$ outputs *failure* and aborts.
Otherwise we have that $x' = x^*$ and $A' \neq A^*$. This means $\mathcal{A}$ can get a new valid SDH pair with $g_1'^{\frac{1}{(x'+\gamma)}}$. Let us show how to get the new SDH pair. Query i^* was chosen uniformly at random and independent from the choices of $\mathcal{F}_2$ among q queries. Therefore, the probability that $x' = x^*$ holds is $1/q$. Therefore, if $x' = x^*$ holds then we get for $v' = s' + m_1'\mu_1 + m_2'\mu_2 + \ldots + m_L'\mu_L$, where $h_i := \psi(u_i)$,$u_i = u_0^{\mu_i}$ for $i = 1, \ldots, L$ and $h_0 = g_1'^{\frac{r^*(x^*+\gamma)-1}{v^*}} = \psi(u_0)$, the following.

$$
\begin{aligned}
A'^{x'+\gamma} &= g_1' \cdot h_0^{v'} \\
A'^{x'+\gamma} &= g_1' \cdot g_1'^{\frac{r^* v'(x^*+\gamma)-v'}{v^*}} && \Leftrightarrow \\
A' &= g_1'^{\frac{1}{x'+\gamma}} \cdot g_1'^{\frac{r^* v'(x^*+\gamma)-v'}{v^*(x'+\gamma)}} \\
&= g_1'^{\frac{1}{x'+\gamma}} \cdot g_1'^{-\frac{v'}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v'}{v^*}} \cdot g_1'^{-\frac{r^* v'(x'-x^*)}{v^*(x'+\gamma)}}, \text{ where } x' = x^* \\
&= g_1'^{\frac{v^*}{v^*(x'+\gamma)}} \cdot g_1'^{-\frac{v'}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v'}{v^*}} \\
&= g_1'^{\frac{v^*-v'}{v^*(x'+\gamma)}} \cdot g_1'^{\frac{r^* v'}{v^*}} && \Leftrightarrow \\
g_1'^{\frac{1}{(x'+\gamma)}} &= \left(A' g_1'^{-\frac{r^* v'}{v^*}} \right)^{\frac{v^*}{v^*-v'}}
\end{aligned}
$$

As in Lemma 4.7, adversary $\mathcal{A}$ is now able to compute and output a solution to the given q-SDH instance. It uses the new q-SDH pair (D', x') where $D' := g_1'^{\frac{1}{(x'+\gamma)}}$ to compute a solution (A, x') to the given q-SDH instance. The process is the same as in Lemma 4.7 and left out here. The result is a SDH pair $(A, x') \in \mathbb{G}_1 \times \mathbb{Z}_p$ where $A = g_1^{\frac{1}{x'+\gamma}}$.
Finally $\mathcal{A}$ returns (A, x') as his solution.

Success probability $\mathcal{A}$ succeeds in breaking the q-SDH Assumption only if $\mathcal{F}_2$ outputs a forgery and $\mathcal{A}$ does not output *failure*. Let us call the latter event *failure*. If $\mathcal{A}$ does not output *failure*, then it perfectly simulates the

Sig-forge$_{\mathcal{F}_2,\Pi_B}(\eta)$ game . Therefore, we get the following success probability.

$$\begin{aligned}\varepsilon_{\mathcal{A}} := \Pr[\mathcal{A} \text{ succeeds}] &= \Pr[\mathcal{A} \text{ succeeds} \mid \text{failure}] \cdot \Pr[\text{failure}] \\ &\quad + \Pr\left[\mathcal{A} \text{ succeeds} \mid \overline{\text{failure}}\right] \cdot \Pr\left[\overline{\text{failure}}\right] \\ &= \Pr\left[\mathcal{A} \text{ succeeds} \mid \overline{\text{failure}}\right] \cdot \Pr\left[\overline{\text{failure}}\right] \\ &= \varepsilon_2 \cdot \frac{1}{q}\end{aligned}$$

Hence, the success probability $\varepsilon_{\mathcal{A}}$ of $\mathcal{A}$ is ε_2/q. Its running time is $t_2 + q\mathcal{O}(1)$. Since, the q-SDH Assumption holds for bilinear group generator $\mathcal{G}$, $\varepsilon_{\mathcal{A}}$ is negligible.

□

Lemma 4.9. *Under the q-SDH Assumption, the success probability of forger $\mathcal{F}_3$ in the Sig-forge$_{\mathcal{F}_3,\Pi_B}(\eta)$ game is negligible.*

Proof. First assume that the success probability of $\mathcal{F}_3$ is ε_3 in the Sig-forge$_{\mathcal{F}_2,\Pi_B}(\eta)$ game and its running time is t_3. We construct an algorithm $\mathcal{A}$ that solves the discrete logarithm problem in $\mathbb{G}_1$. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$.

Key Generation: $\mathcal{A}$ generates the key for $\mathcal{F}_3$ as follows:

1. $\mathcal{A}$ chooses $u_i \leftarrow \mathbb{Z}_p$ for $i = 0, \ldots, L$.
2. Chooses $\gamma \leftarrow \mathbb{Z}_p$ and sets $w := g_2^\gamma$
3. Sets his secret key $sk := \gamma$ and initializes $\mathcal{F}_3$ with public key $pk := (g_1, g_2, u_0, \ldots, u_L, w, L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$

Signing: $\mathcal{A}$ answers the signature queries of $\mathcal{F}_3$ as follows. Let $M_i = m_{i,1}, \ldots, m_{i,L}$ be the message block of the i-th query.

1. Chooses $x_i \leftarrow \mathbb{Z}_p$ and $s_i \leftarrow \mathbb{Z}_p$
2. Sets $h_i := \psi(u_i)$ for $i = 0, \ldots, L$.
3. Sets $A_i := (g_1 \cdot h_0^{s_i} \cdot h_1^{m_{i,1}} \cdot \ldots \cdot h_L^{m_L})^{\frac{1}{x_i+\gamma}}$
4. Outputs $\sigma_i := (A_i, x_i, s_I)$

Processing the Forgery Eventually $\mathcal{F}_3$ will output a forgery $\sigma' = (A', x', s')$ on message block M'. Since $\mathcal{F}_3$ only outputs Type 3 forgeries it holds that $x' \in \{x_1, \ldots, x_q\}$ ($x' = x_k$ for some $k \in \{1, \ldots, q\}$) and $A' = A_k$. Since

forgery $\sigma' = (A', x', s')$ on message block M' is valid, we know from the verification check that the following holds:

$$\begin{aligned} e(A'^{x'+\gamma} \cdot h_0^{-s'} \cdot h_1^{-m'_1} \cdot h_2^{-m'_2} \cdot \ldots \cdot h_L^{-m'_L}, g'_2) &= e(g'_1, g'_2) \quad \Leftrightarrow \\ A'^{x'+\gamma} \cdot h_0^{-s'} \cdot h_1^{-m'_1} \cdot h_2^{-m'_2} \cdot \ldots \cdot h_L^{-m'_L} &= g'_1 \quad \Leftrightarrow \\ h_0^{s'} \cdot h_1^{m'_1} \cdot h_2^{m'_2} \cdot \ldots \cdot h_L^{m'_L} &= A'^{x'+\gamma} \cdot g_1'^{-1} \end{aligned}$$

The same holds for $\sigma_k = (A_k, x_k, s_k)$ and $M_k = m_{k,1}, \ldots, m_{k,L}$. Since $x' = x_k$ and $A' = A_k$ it holds that $A'^{x'+\gamma} \cdot g_1'^{-1} = A_k^{x'+\gamma} \cdot g_1'^{-1}$. Therefore, the following equality $h_0^{s'} \cdot h_1^{m'_1} \cdot \ldots \cdot h_L^{m'_L} = h_0^{s_k} \cdot h_1^{m_{k,1}} \cdot \ldots \cdot h_L^{m_{k,L}}$ holds. Let $h_i = h_0^{\mu_i}$ for $i = 1, \ldots, L$.
Notice that $M' \neq M_k$ holds. Let us assume the contrary holds. If $M' = M_k$, $x' = x_k$ and $A' = A_k$ it has to hold that $s' \neq s_k$. This is true since $\mathcal{F}_3$ is not allowed to output a forgery σ' on a message M' where $(M', \sigma') \in Q$. Notice Q is the set of tuples containing the queries made by $\mathcal{F}_3$ and the corresponding answers (signatures). But if $M' = M_k$, $x' = x_k$ and $A' = A_k$ holds, then s' is fixed and has to be equal to s. As a result we get that $(M', \sigma') = (M_k, \sigma_k)$ and therefore $(M', \sigma') \in Q$.
With this said we have that $M' \neq M_k$. Thus, for $j = 1, \ldots, L$ where $m_{k,j} - m'_j \neq 0 \mod p$ we get the following.

$$\begin{aligned} 0 &= s' + m'_1\mu_1 + \ldots + m'_L\mu_L - (s_k + m_{k,1}\mu_1 + \ldots + m_{k,L}\mu_L) \Leftrightarrow \\ \mu_j \left(m_{k,j} - m'_j\right) &= s' - s_k + \mu_1 \left(m'_1 - m_{k,1}\right) + \ldots + \mu_{j-1} \left(m'_{j-1} - m_{k,j-1}\right) + \\ &\quad \mu_{j+1} \left(m'_{j+1} - m_{k,j+1}\right) + \ldots + \mu_L \left(m'_L - m_{i,L}\right) \Leftrightarrow \\ \mu_j &= \frac{s' - s_k + \mu_1 \left(m'_1 - m_{k,1}\right) + \ldots + \mu_{j-1} \left(m'_{j-1} - m_{k,j-1}\right)}{\left(m_{k,j} - m'_j\right)} + \\ &\quad \frac{\mu_{j+1} \left(m'_{j+1} - m_{k,j+1}\right) + \ldots + \mu_L \left(m'_L - m_{i,L}\right)}{\left(m_{k,j} - m'_j\right)} \end{aligned}$$

This is the discrete logarithm of h_j to base h_0 ($\log_{h_0}(h_j)$).

Success probability First, observe that the signature queries are answered by $\mathcal{A}$ as in the real game. Therefore $\mathcal{A}$'s simulation is perfect. Further, $\mathcal{A}$ succeeds in solving the discrete logarithm problem in $\mathbb{G}_1$, if $\mathcal{F}_3$ outputs a forgery. Hence, the success probability of $\mathcal{A}$ is ε_3. Its running time is $t_3 + q\mathcal{O}(1)$. Since, the discrete logarithm problem is assumed to be hard in $\mathbb{G}_1$ the success probability of $\mathcal{A}$ is negligible, where $\mathbb{G}_1$ is an output of the bilinear group generator $\mathcal{G}$.

□

Theorem 4.10. *Under the q-SDH Assumption for bilinear group generator $\mathcal{G}$, the* BBS-B *signature scheme is secure against existential forgery under an adaptive chosen-message attack.*

Proof. Using Lemma 4.7, Lemma 4.8 and Lemma 4.9 we can conclude that if forger $\mathcal{F}$ outputs a forgery for our signature scheme BBS-B in time t with probability $\varepsilon_{\mathcal{F}}$, then the q-SDH Assumption for $\mathcal{G}$ can be broken in time $t+q\mathcal{O}(1)$ with probability at least $\varepsilon/3q$. □

4.2.2 Signature Length

A signature of the BBS-A (Definition 4.1) and BBS-B (Definition 4.4) scheme comprise one element of $\mathbb{G}_1$ and two elements of $\mathbb{Z}_p$. Following the parameters of Boneh et al. [9, 8] one can take p to be a 170-bit prime and use a group $\mathbb{G}_1$, where each group element can be represented by 171 bits. Hence, the total signature length is 511 bits or around 64 bytes. As mentioned in [8] using this parameters, the security is approximately the same as a standard 1024-bit (128 bytes) RSA signature.

In comparison, the signature scheme presented by Boneh et al. [8] achieves a total signature length of 1533 bits or around 192 bytes in the same group setting. More interesting is the comparison with the signature schemes of Camenisch and Lysyanskaya [15] since the schemes are also used to design an anonymous credential system. Their total signature length of the signature scheme for blocks of messages is linear to the size of the blocks. Let the size of a block be B and let us also use a group, where each group element can be represented by 171 bits. Then we get a total signature length of $513+B\cdot 342$ bits or about $64+B\cdot 43$ bytes. Their basic signature scheme has a total signature length of about 919 bits or 115 bytes. Further the signature schemes of Camenisch and Lysyanskaya [15] are presented in a setting where $\mathbb{G}_1 = \mathbb{G}_2$ whereas our signature schemes are presented for the more general case where $\mathbb{G}_1 \neq \mathbb{G}_2$ is allowed.
With long-term security and an efficient implementation in mind, following Barreto and Naehrig [3], one can take an embedding degree of 12. Using the same group setup for $\mathbb{G}_1$ with a 171 bits representation, we achieve for our signature schemes a security that is approximately the same as a 2048-bit RSA signature. As mentioned before, the signature schemes as presented in [15] are defined for the setting where $\mathbb{G}_1 = \mathbb{G}_2$. Following Joye and Neven [33, Chapter 2] the maximum possible embedding degree for this setting is 6. Therefore, to achieve higher security levels for the signature schemes of [15], one have to increase the underlying prime. Thus further increasing the total signature length.

5 Protocols for the Signature Schemes

The signature schemes, presented in the previous Chapter 4, are suitable for the design of efficient protocols. Regarding the design of our anonymous credential system, let us make clear that the protocols are a crucial part for any transaction in our system. The first efficient protocol in this chapter will be a protocol for obtaining a signature on a committed value. A user who has formed a pseudonym (resp. commitment) with an organization, asks this organization to grant him a credential for his pseudonym. In the process the user has to prove that he possesses (resp. knows) the identity behind the pseudonym (resp. the committed value). Another essential transaction of an anonymous credential system is the demonstration of a credential to another party. A party is either an organization or verifier. Whereas, the latter one does not grant credentials. However a verifier can be a subscription-based service provider, which requires a credential of a specific organization. Therefore, we will present a protocol for proving the knowledge of a signature on a committed value. In other terms, a user proves to the verifier (resp. organization) that he owns a valid credential of an organization. In detail, we will construct the protocol using two other protocols. Namely a protocol for proving knowledge of a signature and a protocol for proving knowledge of a committed value.

5.1 A Protocol for Signing a Committed Value

A user of our anonymous credential system forms a commitment representing his pseudonym. Now, we need a way to issue a credential (resp. signature) on the committed value. A protocol for signing a committed value for the BBS-B signature scheme will be presented in this section.

In anonymous credential systems the credentials are granted by organizations. The following protocol follows the specification presented in Section 3.3. In the following, we will refer to an organization as the signer. The protocol will be used in a transaction of our anonymous credential system to grant a credentials for an established pseudonym. In detail, it will be used in the $\mathit{GrantCred}(U, N_O, O)$ transaction between a user U and organization O, where N_O is an established pseudonym (resp. commitment) of U with O.

Protocol 5.1. *Let* $\Pi_B = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ *be the secure signature scheme of Definition 4.4* (BBS-B)*,* $\mathcal{C}^L = (\mathrm{Gen}_{\mathcal{C}}, \mathrm{Commit})$ *the commitment scheme of Defini-*

tion 2.10 and let the zero-knowledge proof of knowledge be the one obtained from Protocol 2.25 with the technique by Damgård [24]. Further, let $pk = (g_1, g_2, u_0, u_1, \ldots, u_L, w, L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$*,* $sk := \gamma$ *the public and secret key of* Π_B*, where* $(pk, sk) \leftarrow \mathrm{Gen}(1^\eta)$ *and the commitment public key be* $pk_{\mathcal{C}} = (u_0, \ldots, u_L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$*, where* $pk_{\mathcal{C}}$ *is part of* pk*. Let* $h_i := \psi(u_i)$ *for* $i = 0, \ldots, L$*. The protocol for signing a committed value is defined as follows.*

Common Input *The public key* $pk = (g_1, g_2, u_0, \ldots, u_L, w, L, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$ *and a commitment* C*.*

User's private Input *Values* $s', m_1, \ldots, m_L \in \mathbb{Z}_p$ *such that* $C = h_0^{s'} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} = \mathrm{Commit}\left(pk_{\mathcal{C}}, (m_1, \ldots, m_L), s'\right)$*.*

Signer's private Input *Secret key* $sk = \gamma$*.*

Protocol *The user gives a ZKPK of the opening of the commitment*

$$ZKPK\left\{(s', m_1, \ldots, m_L) : C = h_0^{s'} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}\right\}$$

If the signer accepts the ZKPK, then it executes the following steps. Otherwise it aborts.

1. *Chooses* $x \leftarrow \mathbb{Z}_p$ *and* $s'' \leftarrow \mathbb{Z}_p$
2. $A = (g_1 \cdot h_0^{s''} \cdot C)^{\frac{1}{x+\gamma}}$
3. *Sends* (A, x, s'') *to the user*

The user sets $s = s' + s'' \mod p$ *and sets the signature* σ *on the block of messages* $M = (m_1, \ldots, m_L)$ *to* $\sigma := (A, x, s)$*.*

In the setting of an anonymous credential system the resulting signature σ is called a credential. The credential is not published and only known to the user participating in the protocol.

Let us show that the output of the user is a correct signature on the block of messages of length L under the secret key sk.

Theorem 5.2. *Protocol 5.1 for signing a committed value is correct.*

Proof. Let us show that, for every key pair $(pk, sk) \leftarrow \mathrm{Gen}(1^\eta)$ and values $m_1, \ldots, m_L$, $s' \in \mathbb{Z}_p$, such that $C = h_0^{s'} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$, $\mathrm{Verify}(pk, M, \sigma)$ of scheme Π_B outputs 1 after an execution of the Protocol 5.1 with $\sigma = (A, x, s)$ as the user's output. For $\sigma = (A, x, s)$ it holds that $A = (g_1 \cdot h_0^{s''} \cdot C)^{\frac{1}{x+\gamma}}$, $A \in \mathbb{G}_1$, $x \in \mathbb{Z}_p$ and

$s = s' + s'' \mod p$. For the check of Verify(pk, M, σ) we get the following:

$$\begin{aligned}
& e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(A, g_2^x) \cdot e(A, g_2^\gamma) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(A, g_2^{x+\gamma}) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e((g_1 \cdot h_0^{s''} \cdot C)^{\frac{x+\gamma}{x+\gamma}} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(g_1 \cdot h_0^{s'} \cdot h_0^{s''} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} \cdot h_0^{-s'} \cdot h_0^{-s''} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) \\
\Leftrightarrow\ & e(g_1, g_2)
\end{aligned}$$

Hence, Verify (pk, M, σ) outputs 1. □

Theorem 5.3. *Protocol 5.1 is a secure two-party protocol in terms of Definition 3.8 for signing a committed value.*

Proof. First, notice that Protocol 5.1 follows the functional specification of Definition 3.7. We will prove the security of the protocol for signing a committed value according to Definition 3.8. Following Definition 3.8 the proof will be split into two parts. First we look at the security for the signer and then at the security for the user. Protocol 5.1 is constructed using the signature scheme BBS-B of Definition 4.4, the commitment scheme of Definition 2.10 and the *ZKPK* obtained from Protocol 2.25 with the standard technique by Damgård [24]. In the following let $(pk, sk) \leftarrow \text{Gen}(1^\eta)$. Let us first prove the security for the signer in Protocol 5.1.

Security for the signer Since the protocol uses a *ZKPK* to prove the knowledge of the opening of the commitment, there is an extractor that can extract the opening (respectively committed value). Let us call the extractor $\mathcal{E}$. $\mathcal{E}$ is now given single oracle access to Sign$(pk, sk, \cdot)$ of the signature scheme BBS-B from Definition 4.4. Let U be a user participating in Protocol 5.1. Let $m_1, \ldots, m_L, s' \in \mathbb{Z}_p$ be U's private input, such that $C = h_0^{s'} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$. Then, the extractor $\mathcal{E}$ with black-box access to U $(\mathcal{E}^{1\,\text{Sign}(pk,sk,\cdot)}(1^\eta) \xrightarrow{\blacksquare} U_\eta(pk))$ can extract the values $s', m_1, \ldots, m_L$ of the commitment $C = h_0^{s'} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$. Next $\mathcal{E}$ queries his oracle as Sign(pk, sk, M), where $M = m_1, \ldots, m_L$. The output $\sigma' = (A, x, r)$ of the oracle is a valid signature under the signatures secret key sk. Since the oracle works as the signing algorithm of Definition 4.4, we have that $x, r \leftarrow \mathbb{Z}_p$, $A = (g_1 \cdot h_0^r \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L})^{\frac{1}{x+\gamma}}$ and $e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-r} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2)$ $= e(g_1, g_2)$ holds. The extractor $\mathcal{E}$ sets $s'' = r - s' \mod p$ and outputs (A, x, s'') to U. The last step of the protocol is that U sets $s = s' + s'' = s' + r - s' = r \mod p$. Since r is the output of the oracle it holds that $r \leftarrow \mathbb{Z}_p$. This implies that s is uniformly distributed in $\mathbb{Z}_p$. Observe, since $s = r$ it directly holds that Verify$(pk, M, (A, x, s)) = 1$, i.e. (A, x, s) is a valid signature on M under the secret key sk.

Let us now show that in the protocol $\mathcal{E}^{1\,\mathrm{Sign}(pk,sk,\cdot)}(1^\eta) \overset{\blacksquare}{\to} U_\eta(pk)$ the tuples (A, x, s) and (A, x, s'') are distributed as in the real protocol $O(sk) \leftrightarrow U_\eta(pk)$. In the real protocol, U interacts with the signer O and sets $s = s' + s''$ mod p. For correctly generated commitment C it holds that s' was chosen uniformly at random from $\mathbb{Z}_p$. Further, the signer O in the real protocol chooses $s'' \leftarrow \mathbb{Z}_p$. This implies that s is uniformly distributed in $\mathbb{Z}_p$, both in the protocol with $\mathcal{E}$ and in the real protocol with O. Furthermore, $\mathcal{E}$'s output values A, x and s'' are distributed as in the real protocol since they are the output of the oracle $\mathrm{Sign}(pk, sk, \cdot)$. Therefore, (A, x, s) and (A, x, s'') are distributed as in the real protocol. In conclusion, a user U cannot distinguish between the interaction with $\mathcal{E}$ and O. Due to the fact that the *ZKPK* remains zero-knowledge under sequential composition, this is true even if we consider sequentially composed signer protocols.

Security for the user Since the commitment scheme $\mathcal{C}^L$ is perfectly hiding, the signer learns nothing from the correctly generated commitment $C = h_0^{s'} \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L} = \mathrm{Commit}\left(pk_{\mathcal{C}}, (m_1, \ldots, m_L), s'\right)$. We assume that s' was chosen uniformly at random from $\mathbb{Z}_p$. Otherwise the user's commitment was not correctly generated. The only other step, where the user's private input $m_1, \ldots, m_L, s' \in \mathbb{Z}_p$ is involved, is the *ZKPK*. Consequently, an adversarial signer has to gain knowledge about the user's private input through the *ZKPK*. With that said, assume there is an ppt adversary $\mathcal{A}$ such that for $X_1 = (m_{1,1}, \ldots, m_{1,L}) \in \mathbb{Z}_p^L$, $X_2 = (m_{2,1}, \ldots, m_{2,L}) \in \mathbb{Z}_p^L$ and for $pk \in \mathrm{Gen}(1^\eta)$ (pk is known to everyone),

$$\left| \Pr\left[s_1' \leftarrow \mathbb{Z}_p, b \leftarrow (\mathcal{A} \leftrightarrow U\left(X_1, s_1'\right)) : b = 1\right] - \Pr\left[s_2' \leftarrow \mathbb{Z}_p, b \leftarrow (\mathcal{A} \leftrightarrow U\left(X_2, s_2'\right)) : b = 1\right] \right| = \nu(\eta)$$

is not a negligible function in η. Notice that *ZKPK* protocols are also protocols with witness indistinguishability. But $\mathcal{A}$ is are able to distinguish the witnesses (X_1, s_1') and (X_2, s_2'). This contradicts the zero-knowledge property of the used protocol. Thus, $\nu(\eta)$ is negligible in η. In conclusion, the only thing that an adversary learns by interacting with U is the commitment C.

□

5.1.1 Efficiency

We analyze the total communicated data of Protocol 5.1, using the same group setup as in 4.2.2. Notice, we will analyze the sub-protocol as the *HVZK* protocol presented in Protocol 2.25. Thus, we ignore the constant overhead related

to the transformation to a zero-knowledge proof of knowledge with the standard technique by Damgård [24]. Hence, in Protocol 2.25 there are $341 + L \cdot 170$ bits exchanged. The additional communicated data in Protocol 5.1 is that the signer sends $(A, x, s'') \in \mathbb{G}_1 \times \mathbb{Z}_p^2$ to the user or 511 bits. In total, the communicated data is $852 + L \cdot 170$ bits or around $107 + L \cdot 22$ bytes.

In the next two sections, we want to present a protocol for proving knowledge of a signature an a committed value. This protocol will be used, by a user in our anonymous credential system, to proof the possession of a credential of a specific organization to another organization, who knows the user under a different pseudonym than the issuing organization. Where the protocol ensures that the credential was issued to the user that formed both pseudonyms. The protocol itself will be a combination of the protocol for proving knowledge of a committed value (Protocol 2.25) and the next protocol.

5.2 A Protocol for Proving Knowledge of a Signature

The protocol presented in this section proves the possession of a valid signature on a block of messages from the BBS-B signature scheme (Definition 4.4). The protocol can also be used for the BBS-A signature scheme of Definition 4.1. The protocol for BBS-B is later used as a part of the protocol for proving knowledge of a signature on a committed value in Section 5.3. The protocol is also of interest to design other systems than anonymous credential systems. A basic form of the protocol is originally presented in [8] and is extended in [5]. Where in the latter one it is used in the design of a reputation system. We will extend the basic protocol such that it suits our needs for a block signature. This variation of the protocol is also mentioned in [8]. A more efficient variant of the protocol will be presented in Section 6.3.2. Let us give the formal definition of the protocol in the following.

Protocol 5.4. *Let $\mathcal{G}$ be a bilinear group generator and $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$ such that $g_1 = \psi(g_2)$. Further, let $\Pi_B = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ be the secure signature scheme of Definition 4.4* (BBS-B) *and $(pk, sk) \leftarrow \mathrm{Gen}(1^\eta)$, where $pk := (g_1, g_2, u_0, \ldots, u_L, w)$. The protocol proves the possession of the elements $\sigma = (A, x, s) \in \mathbb{G}_1 \times \mathbb{Z}_p^2$ and $m_1, \ldots, m_L \in \mathbb{Z}_p^L$ such that $\mathrm{Verify}(pk, (m_1, \ldots m_L), \sigma) = 1$. The value A is uniquely determined by $A = (g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L})^{\frac{1}{x+\gamma}}$, where the value $\gamma \in \mathbb{Z}_p$ is secret and not known to the prover. Following the verification check of $\mathrm{Verify}(pk, (m_1, \ldots m_L), \sigma)$, it holds that $e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) = e(g_1, g_2)$. Notice that we exclude the run of the bilinear group generator from* Gen. *In our anonymous credential system we will rely on a separate initialization algorithm using a bilinear group generator $\mathcal{G}$ for the setup of $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi,$*

e). The formal outline of the protocol for proving knowledge of a signature on a block of messages is defined as follows.

Common Input *The signature public key* $pk = (g_1, g_2, u_0, \ldots, u_L, w)$ *and* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$. *Additional public values are* $g_1, h_0, \ldots, h_L, u, v, z \in \mathbb{G}_1$, $g_2, w \in \mathbb{G}_2$ *and* L, *where* L *has polynomial length in* η, $g_2, u_0, \ldots, u_L \leftarrow \mathbb{G}_2$, $u, v, z \leftarrow \mathbb{G}_1$, $h_i = \psi(u_i)$ *for* $i = 0, \ldots, L$, $w = g_2^{\gamma}$ *and* $g_1 = \psi(g_2)$

User's private Input *Values* $A \in \mathbb{G}_1$ *and* $x, s, m_1, \ldots, m_L \in \mathbb{Z}_p$ *where* $\sigma = (A, x, s)$ *such that*

$$\mathrm{Verify}(pk, (m_1, \ldots, m_L), \sigma) = 1$$

Protocol

1. *Prover* P *selects exponents* $\alpha, \beta \leftarrow \mathbb{Z}_p$ *and computes a Linear Encryption*

$$T_1 := u^{\alpha} \qquad T_2 := v^{\beta} \qquad T_3 := Az^{\alpha+\beta}$$

   P *also computes two helper values:*

$$\delta_1 := x\alpha \qquad \delta_2 := x\beta$$

   P *and* V *then undertake a proof of knowledge of values* $(\alpha, \beta, x, s, m_1, \ldots, m_L, \delta_1, \delta_2)$.

2. P *chooses blinding values* $r_\alpha, r_\beta, r_x, r_s, r_{m_1}, \ldots, r_{m_L}, r_{\delta_1}, r_{\delta_2} \leftarrow \mathbb{Z}_p$ *and computes:*

$$R_3 := e(T_3, g_2)^{r_x} \cdot e(z, w)^{-r_\alpha - r_\beta} \cdot e(z, g_2)^{-r_{\delta_1} - r_{\delta_2}} \cdot e(h_0, g_2)^{-r_s} \cdot \prod_{i=1}^{L} \cdot e(h_i, g_2)^{-r_{m_i}}$$

$$R_1 := u^{r_\alpha} \qquad R_2 := v^{r_\beta}$$
$$R_4 := T_1^{r_x} \cdot u^{-r_{\delta_1}} \qquad R_5 := T_2^{r_x} \cdot v^{-r_{\delta_2}}$$

   and sends $(T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)$ *to* V.

3. V *chooses* $c \leftarrow \mathbb{Z}_p$ *and sends it to* P.

4. P *computes and sends back to* V:

$$s_\alpha = r_\alpha + c \cdot \alpha \qquad s_\beta = r_\beta + c \cdot \beta$$
$$s_{\delta_1} = r_{\delta_1} + c \cdot \delta_1 \qquad s_{\delta_2} = r_{\delta_2} + c \cdot \delta_2$$
$$s_x = r_x + c \cdot x \qquad s_{m_i} = r_{m_i} + c \cdot m_i \text{ , for } i = 1, \ldots L$$
$$s_s = r_s + c \cdot s$$

5. *V verifies that the following holds:*

$$
\begin{aligned}
R_1 &= u^{s_\alpha} \cdot T_1^{-c} && (5.1)\\
R_2 &= v^{s_\beta} \cdot T_2^{-c} && (5.2)\\
R_3 &= e(T_3, g_2)^{s_x} \cdot e(z, w)^{-s_\alpha - s_\beta} \cdot e(z, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot e(h_0, g_2)^{-s_s} \cdot && (5.3)\\
&\quad e(h_1, g_2)^{-s_{m_1}} \cdot \ldots \cdot e(h_L, g_2)^{-s_{m_L}} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^c \\
R_4 &= T_1^{s_x} \cdot u^{-s_{\delta_1}} && (5.4)\\
R_5 &= T_2^{s_x} \cdot v^{-s_{\delta_2}} && (5.5)
\end{aligned}
$$

6. *V accepts if all checks are satisfied.*

Theorem 5.5. *Protocol 5.4 is a Σ-protocol for proving knowledge of the values $(\alpha, \beta, x, s, m_1, \ldots, m_L, \delta_1, \delta_2)$ such that $T_1 = u^\alpha \wedge T_2 = v^\beta \wedge \frac{e(T_3,w)}{e(g_1,g_2)} = e(T_3, g_2)^{-x} \cdot e(z, w)^{\alpha+\beta} \cdot e(z, g_2)^{\delta_1+\delta_2} \cdot e(h_0, g_2)^{s'} \cdot e(h_1, g_2)^{m_1} \cdot \ldots \cdot e(h_L, g_2)^{m_L} \wedge T_1^x = u^{\delta_1} \wedge T_2^x = v^{\delta_2}$.*

Proof. The theorem follows from the three-round form of the presented protocol, the completeness (Lemma 5.6), SHVZK property (Lemma 5.7) and special soundness property (Lemma 5.8). □

First we show that the protocol is complete; second, the protocol is a special honest-verifier zero-knowledge protocol; and third, the protocol satisfies the special soundness property.

Lemma 5.6. *Protocol 5.4 is complete.*

Proof. Assume P is a honest prover in possession of $(A, x, s, m_1, \ldots, m_L)$, this means $A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L} = g_1$. P follows the specified steps of the protocol and therefore we get the following for the verification equations:

$$
\begin{aligned}
u^{s_\alpha} \cdot T_1^{-c} &= u^{r_\alpha + c\alpha} \cdot u^{-c\alpha} = u^{r_\alpha} = R_1\\
v^{s_\beta} \cdot T_2^{-c} &= v^{r_\beta + c\beta} \cdot v^{-c\beta} = v^{r_\beta} = R_2
\end{aligned}
$$

$$
\begin{aligned}
T_1^{s_x} \cdot u^{-s_{\delta_1}} &= u^{\alpha(r_x + cx)} \cdot u^{-r_{\delta_1} - \alpha cx} = T_1^{r_x} \cdot u^{-r_{\delta_1}} = R_4\\
T_2^{s_x} \cdot v^{-s_{\delta_2}} &= v^{\beta(r_x + cx)} \cdot v^{-r_{\delta_2} - \beta cx} = T_2^{r_x} \cdot v^{-r_{\delta_2}} = R_5
\end{aligned}
$$

For R_3 we get the following:

$$
\begin{aligned}
& e(T_3, g_2)^{s_x} \cdot e(z, w)^{-s_\alpha - s_\beta} \cdot e(z, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot e(h_0, g_2)^{-s_s} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-s_{m_i}} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^c \\
=& e(T_3, g_2)^{r_x + cx} \cdot e(z, w)^{-r_\alpha - c\alpha - r_\beta - c\beta} \cdot e(z, g_2)^{-r_{\delta_1} - c\delta_1 - r_{\delta_2} - c\delta_2} \cdot e(h_0, g_2)^{-r_s - cs} \cdot \\
& \prod_{i=1}^{L} e(h_i, g_2)^{-r_{m_i} - cm_i} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^c \\
=& \frac{e(T_3, g_2)^{cx} \cdot e(z, w)^{-c\alpha - c\beta} \cdot e(z, g_2)^{-c\delta_1 - c\delta_2} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_3}{e(T_3, w)^{-c} e(g_1, g_2)^c} \\
=& \frac{e(Az^{\alpha+\beta}, g_2)^{cx} \cdot e(z, w)^{-c\alpha - c\beta} \cdot e(z, g_2)^{cx \cdot (-\alpha - \beta)} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_3}{e(T_3, w)^{-c} e(g_1, g_2)^c} \\
=& \frac{e(Az^{\alpha+\beta}, g_2)^{cx} \cdot e(z, w)^{-c\alpha - c\beta} \cdot e(z^{-\alpha-\beta}, g_2)^{cx} \cdot e(h_0, g_2)^{-cs} \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_3}{e(T_3, w)^{-c} e(g_1, g_2)^c} \\
=& e(A, g_2)^{cx} e(z, w)^{-c\alpha - c\beta} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_3 \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^c \\
=& e(A, g_2)^{cx} e(z, w)^{c \cdot (-\alpha - \beta)} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_3 \cdot \left(\frac{e(Az^{\alpha+\beta}, w)}{e(g_1, g_2)}\right)^c \\
=& e(A, g_2)^{cx} e(A, w)^c \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_3 \cdot \left(\frac{1}{e(g_1, g_2)}\right)^c \\
=& e(A^x, g_2)^c e(A^\gamma, g_2)^c \cdot e(h_0^{-s}, g_2)^c \cdot \prod_{i=1}^{L} e(h_i^{m_i}, g_2)^{-c} \cdot R_3 \cdot \left(\frac{1}{e(g_1, g_2)}\right)^c \\
=& R_3 \cdot \left(\frac{e(A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2)}{e(g_1, g_2)}\right)^c \\
=& R_3 \cdot \left(\frac{e(g_1, g_2)}{e(g_1, g_2)}\right)^c \\
=& R_3
\end{aligned}
$$

All verification equations are satisfied. Therefore, verifier V will always accept if prover P is honest. □

Lemma 5.7. *Protocol 5.4 is a (computational) special honest-verifier zero-knowledge (SHVZK) protocol under the assumption that the Decision Linear Problem holds in $\mathbb{G}_1$.*

Proof. We describe a simulator $\mathcal{S}$ that outputs transcripts of Protocol 5.4. $\mathcal{S}$ on input the public values of the protocol and a challenge $c' \in \mathbb{Z}_p$, chooses $s', m'_1, \ldots, m'_L, \alpha', \beta' \leftarrow \mathbb{Z}_p$, $A' \leftarrow \mathbb{G}_1$ and sets:

$$T_1 = u^{\alpha'} \qquad T_2 = v^{\beta'} \qquad T_3 = A' z^{\alpha' + \beta'}$$

$\mathcal{S}$ chooses $s_{\alpha'}, s_{\beta'}, s_{x'}, s_{s'}, s_{m'_1}, \ldots, s_{m'_L}, s_{\delta'_1}, s_{\delta'_2} \leftarrow \mathbb{Z}_p$, and sets

$$\begin{aligned}
R_1 &:= u^{s_{\alpha'}} \cdot T_1^{-c'} \\
R_2 &:= v^{s_{\beta'}} \cdot T_2^{-c'} \\
R_3 &:= e(T_3, g_2)^{s_{x'}} \cdot e(z, w)^{-s_{\alpha'} - s_{\beta'}} \cdot e(z, g_2)^{-s_{\delta'_1} - s_{\delta'_2}} \cdot e(h_0, g_2)^{-s'} \cdot \\
&\quad e(h_1, g_2)^{-s_{m'_1}} \cdot \ldots \cdot e(h_L, g_2)^{-s_{m'_L}} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^{c'} \\
R_4 &:= T_1^{s_{x'}} \cdot u^{-s_{\delta'_1}} \\
R_5 &:= T_2^{s_{x'}} \cdot v^{-s_{\delta'_2}}
\end{aligned}$$

It outputs the transcript (T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5, c', $s_{\alpha'}$, $s_{\beta'}$, $s_{x'}$, $s_{s'}$, $s_{m'_1}$, $\ldots, s_{m'_L}$, $s_{\delta'_1}$, $s_{\delta'_2}$). The verification equations are satisfied because of the following. For given challenge c', after choosing $s_{\alpha'}$, $s_{\beta'}$, $s_{x'}$, $s_{s'}$, $s_{m'_1}$, $\ldots, s_{m'_L}$, $s_{\delta'_1}$, $s_{\delta'_2}$ uniformly at random from $\mathbb{Z}_p$ the values R_1, R_2, R_3, R_4, R_5 are fixed, such that the verification equations are directly satisfied. With public values u, v, z the tuple (u, v, z, T_1, T_2, T_3) is an instance of the Decision Linear Problem in $\mathbb{G}_1$. The instance is completely random and thus the distribution of the tuple in the simulation can not be distinguished from the distribution in the real protocol. Overall, the tuple (R_1, R_2, R_3, R_4, R_5, c', $s_{\alpha'}$, $s_{\beta'}$, $s_{x'}$, $s_{s'}$, $s_{m'_1}$, $\ldots, s_{m'_L}$, $s_{\delta'_1}$, $s_{\delta'_2}$) is distributed as in the real protocol. Assuming the Decision Linear Problem holds in $\mathbb{G}_1$ and using a standard hybrid argument, it follows that the transcripts (T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5, c', $s_{\alpha'}$, $s_{\beta'}$, $s_{x'}$, $s_{s'}$, $s_{m'_1}$, $\ldots, s_{m_L}$, $s_{\delta'_1}$, $s_{\delta'_2}$) of $\mathcal{S}$ are indistinguishable from the transcripts of the real protocol. □

Lemma 5.8. *Protocol 5.4 satisfies the special soundness property.*

Proof. We show that there is an extractor $\mathcal{E}$ that can extract a witness given two accepting transcripts of Protocol 5.4. Let the transcripts be (T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5, c, s_α, s_β, s_x, s_s, s_{m_1}, $\ldots, s_{m_L}$, s_{δ_1}, s_{δ_2}) with challenge c and (T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5, c', s'_α, s'_β, s'_x, s'_s, s'_{m_1}, $\ldots, s'_{m_L}$, s'_{δ_1}, s'_{δ_2}) with challenge c', where $c \neq c'$. $\mathcal{E}$ extracts the witness as follows. First, it sets the following helper values:

$$\begin{aligned}
\Delta c &:= c - c' & \Delta s_\alpha &:= s_\alpha - s'_\alpha & \Delta s_\beta &:= s_\beta - s'_\beta \\
\Delta s_x &:= s_x - s'_x & \Delta s_s &:= s_s - s'_s & \Delta s_{m_i} &:= s_{m_i} - s'_{m_i}\text{, for } i = 1, \ldots, L \\
\Delta s_{\delta_1} &:= s_{\delta_1} - s'_{\delta_1} & \Delta s_{\delta_2} &:= s_{\delta_2} - s'_{\delta_2}
\end{aligned}$$

$$\begin{aligned}
\tilde{\alpha} &:= \frac{\Delta s_\alpha}{\Delta c} & \tilde{\beta} &:= \frac{\Delta s_\beta}{\Delta c} & \tilde{m}_i &:= \frac{\Delta s_{m_i}}{\Delta c}\text{, for } i = 1, \ldots, L \\
\tilde{x} &:= \frac{\Delta s_x}{\Delta c} & \tilde{s} &:= \frac{\Delta s_s}{\Delta c}
\end{aligned}$$

Following the verification checks and dividing both instances we get:

$$u^{s_\alpha} \cdot T_1^{-c} \cdot u^{-s'_\alpha} \cdot T_1^{c'} = u^{s_\alpha - s'_\alpha} \cdot T_1^{c'-c} = u^{\Delta s_\alpha} \cdot T_1^{-\Delta c} \Rightarrow T_1^{\Delta c} = u^{\Delta s_\alpha} \Rightarrow T_1 = u^{\tilde{\alpha}}$$
$$v^{s_\beta} \cdot T_2^{-c} \cdot v^{-s'_\beta} \cdot T_2^{c'} = v^{s_\beta - s'_\beta} \cdot T_2^{c'-c} = v^{\Delta s_\beta} \cdot T_2^{-\Delta c} \Rightarrow T_2^{\Delta c} = v^{\Delta s_\beta} \Rightarrow T_2 = v^{\tilde{\beta}}$$

$$T_1^{s_x} \cdot u^{-s_{\delta_1}} \cdot T_1^{-s'_x} \cdot u^{s'_{\delta_1}} = T_1^{s_x - s'_x} \cdot u^{s'_{\delta_1} - s_{\delta_1}} = T_1^{\Delta s_x} \cdot u^{-\Delta s_{\delta_1}}$$
$$\Rightarrow T_1^{\Delta s_x} = u^{\Delta s_{\delta_1}} \Rightarrow \tilde{\alpha} \Delta s_x = \Delta s_{\delta_1}$$

$$T_2^{s_x} \cdot v^{-s_{\delta_2}} \cdot T_2^{-s'_x} \cdot v^{s'_{\delta_2}} = T_2^{s_x - s'_x} \cdot v^{s'_{\delta_2} - s_{\delta_2}} = T_2^{\Delta s_x} \cdot v^{-\Delta s_{\delta_2}}$$
$$\Rightarrow T_2^{\Delta s_x} = v^{\Delta s_{\delta_2}} \Rightarrow \tilde{\beta} \Delta s_x = \Delta s_{\delta_2}$$

With that established, we get by dividing the two instances of the verification equation for R_3 the following:

$$\frac{e(T_3, g_2)^{s_x} \cdot e(z, w)^{-s_\alpha - s_\beta} \cdot e(z, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot e(h_0, g_2)^{-s_s} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-s_{m_i}} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^{c}}{e(T_3, g_2)^{s'_x} \cdot e(z, w)^{-s'_\alpha - s'_\beta} \cdot e(z, g_2)^{-s'_{\delta_1} - s'_{\delta_2}} \cdot e(h_0, g_2)^{-s'_s} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-s'_{m_i}} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^{c'}}$$
$$= e(T_3, g_2)^{\Delta s_x} \cdot e(z, w)^{-\Delta s_\alpha - \Delta s_\beta} \cdot e(z, g_2)^{-\Delta s_{\delta_1} - \Delta s_{\delta_2}} \cdot e(h_0, g_2)^{-\Delta s_s} \cdot$$
$$e(h_1, g_2)^{-\Delta s_{m_1}} \cdot \ldots \cdot e(h_L, g_2)^{-\Delta s_{m_L}} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^{\Delta c}$$

This is equivalent to

$$\left(\frac{e(g_1, g_2)}{e(T_3, w)}\right)^{\Delta c} = e(T_3, g_2)^{\Delta s_x} \cdot e(z, w)^{-\Delta s_\alpha - \Delta s_\beta} \cdot e(v, g_2)^{-\tilde{\alpha} \Delta s_x - \tilde{\beta} \Delta s_x} \cdot e(h_0, g_2)^{-\Delta s_s} \cdot$$
$$e(h_1, g_2)^{-\Delta s_{m_1}} \cdot \ldots \cdot e(h_L, g_2)^{-\Delta s_{m_L}} \qquad \Leftrightarrow$$
$$\left(\frac{e(g_1, g_2)}{e(T_3, w)}\right) = e(T_3, g_2)^{\tilde{x}} \cdot e(z, w)^{-\tilde{\alpha} - \tilde{\beta}} \cdot e(z, g_2)^{-\tilde{x}(\tilde{\alpha} + \tilde{\beta})} \cdot e(h_0, g_2)^{-\tilde{s}} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-\tilde{m}_i}$$
$$e(g_1, g_2) = e(T_3, g_2)^{\tilde{x}} \cdot e(z^{-\tilde{\alpha} - \tilde{\beta}}, w) \cdot e(z^{-\tilde{\alpha} - \tilde{\beta}}, g_2)^{\tilde{x}} \cdot e(h_0, g_2)^{-\tilde{s}} \cdot$$
$$e(h_1, g_2)^{-\tilde{m}_1} \cdot \ldots \cdot e(h_L, g_2)^{-\tilde{m}_L} \cdot e(T_3, w) \qquad \Leftrightarrow$$
$$e(g_1, g_2) = e(T_3 z^{-\tilde{\alpha} - \tilde{\beta}}, g_2)^{\tilde{x}} \cdot e(T_3 z^{-\tilde{\alpha} - \tilde{\beta}}, w) \cdot e(h_0, g_2)^{-\tilde{s}} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-\tilde{m}_i}$$

with $\tilde{A} = T_3 z^{-\tilde{\alpha} - \tilde{\beta}}$ we get:

$$e(g_1, g_2) = e(\tilde{A}, g_2)^{\tilde{x}} \cdot e(\tilde{A}, w) \cdot e(h_0, g_2)^{-\tilde{s}} \cdot e(h_1, g_2)^{-\tilde{m}_1} \cdot \ldots \cdot e(h_L, g_2)^{-\tilde{m}_L}$$

The result $(\tilde{A}, \tilde{x}, \tilde{s}, \tilde{m}_1, \ldots, \tilde{m}_L)$ is a valid signature $\tilde{\sigma} = (\tilde{A}, \tilde{x}, \tilde{s})$ on the block of messages $\tilde{m}_1, \ldots, \tilde{m}_L$. Moreover the $\tilde{A}$ is perforce the same as in the Linear

Encryption (T_1, T_2, T_3). Observe, that $\mathcal{E}$ runs in polynomial-time since it only divides the instances of the verification equations. □

5.2.1 Efficiency

We want to analyze the amount of data that is exchanged between the prover P and verifier V in the protocol above. As mentioned before in 4.2.2 we follow the parameters of Boneh et al. [9, 8] and take p to be a 170-bit prime, use a group $\mathbb{G}_1$, where each group element can be represented by 171 bits, and use an embedding degree of 6. Then, the total communicated data in the protocol above is $3407 + L \cdot 170$ bits or around $426 + L \cdot 22$ bytes.

With the above Protocol 5.4 we are now able to proof the possession of a valid signature that was issued using the BBS-B signature scheme. For the design of our anonymous credential system we want to use it as a part of a another protocol. Suppose a organization O_1 knows a user by the pseudonym N_{O_1} and has granted a credential for the pseudonym. We want a protocol that allows a user to proof the possession of the credential to another organization O_2. Where O_2 knows the user under a different pseudonym N_{O_2} than the credential issuing organization. Therefore the protocol has to proof that the credential is granted to a pseudonym that belongs to the same user that has formed N_{O_2}. Or in other terms, that the signature was issued on one and the same committed value. Next, we will show this protocol in detail.

5.3 A Protocol for Proving Knowledge of a Signature on a Committed Value

The protocol itself is a combination of the protocol for proving knowledge of a committed value Protocol 2.25 and the protocol for proving knowledge of a signature Protocol 5.4. It will be used in the design of our anonymous credential system as a part of the *VerifyCred*(U, V, N_O, O) and *VerifyCredOnNym*(U, V, N_V, N_O, O) transactions, see Section 6.2. Thus, it is used by a user U to prove the possession of a credential previously granted by organization O to a pseudonym N_O. A part of the protocol is also the proof that the committed value is the same in the given commitment and in the signature, and that the user knows the committed value. Notice that in the following the indices O and V refers to an organization and to a verifier of an anonymous credential system. Where an organization O is a credential granting party and a verifier only verifies existing credentials. Let present the formal definition of the protocol in the following.

Protocol 5.9. *Let $\mathcal{G}$ be a bilinear group generator and* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$. *Let g_1 be a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$ such that $g_1 = \psi(g_2)$. Let $\Pi_{B_O} = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ be the secure signature scheme* BBS-B

of Definition 4.4, $\mathcal{C}_V^L = (\mathrm{Gen}_{\mathcal{C}_V}, \mathrm{Commit}_V)$ is an instances of the commitment scheme of Definition 2.10. The protocol relies on the Protocol 2.25 and Protocol 5.4. Further, let $pk_O := (g_1, g_2, u_0, \ldots, u_L, w)$, $sk_O := \gamma$ be the public and secret key of Π_{B_O}, where $(pk_O, sk_O) \leftarrow \mathrm{Gen}(1^\eta)$ and $w := g_2^\gamma$. Furthermore, let $pk_{\mathcal{C}_V} = (u'_0, \ldots, u'_L)$ be the public key of $\mathcal{C}_V^L$. Notice that we exclude the run of the bilinear group generator from Gen *and* $\mathrm{Gen}_{\mathcal{C}_V}$. *In our anonymous credential system we will rely on a separate initialization algorithm that sets up ($\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, p, ψ, e) using a bilinear group generator $\mathcal{G}$. The protocol is defined as follows.*

Common Input *The organization public key $pk_O = (g_1, g_2, u_0, \ldots, u_L, w)$, the commitment public key $pk_{\mathcal{C}_V} = (u'_0, \ldots, u'_L)$, a commitment C_V and ($\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$, p, ψ, e). Additional public values are $u, v, z \in \mathbb{G}_1$ and L. Where L has polynomial length in η, $u, v, z, \leftarrow \mathbb{G}_1$. Let $h_i = \psi(u_i)$ and $d_i = \psi(u'_i)$ for $i = 0, \ldots, L$.*

User's private Input *Values $A \in \mathbb{G}_1$, $x, s, r_V, m_1, \ldots, m_L \in \mathbb{Z}_p$ where $\sigma = (A, x, s)$ such that*

$$C_V = d_0^{r_V} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L} = \mathrm{Commit}_V\left(pk_{\mathcal{C}_V}, (m_1, \ldots, m_L), r_V\right)$$

and

$$\mathrm{Verify}(pk_O, (m_1, \ldots, m_L), \sigma) = 1$$

Protocol

1. *Prover P selects exponents $\alpha, \beta \leftarrow \mathbb{Z}_p$ and computes a Linear Encryption*

$$T_1 := u^\alpha \qquad T_2 := v^\beta \qquad T_3 := Az^{\alpha+\beta}$$

P also computes two helper values:

$$\delta_1 := x\alpha \qquad \delta_2 := x\beta$$

P and V then undertake a proof of knowledge of values (α, β, x, r_V, s, m_1, $\ldots, m_L$, δ_1, δ_2).

2. *P chooses blinding values $r_\alpha, r_\beta, r_x, r_s, r_{r_V}, r_{m_1}, \ldots, r_{m_L}, r_{\delta_1}, r_{\delta_2} \leftarrow \mathbb{Z}_p$ and computes:*

$$R_3 := e(T_3, g_2)^{r_x} \cdot e(z, w)^{-r_\alpha - r_\beta} \cdot e(z, g_2)^{-r_{\delta_1} - r_{\delta_2}} \cdot e(h_0, g_2)^{-r_s} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-r_{m_i}}$$

$$R_1 := u^{r_\alpha} \qquad R_2 := v^{r_\beta}$$
$$R_4 := T_1^{r_x} \cdot u^{-r_{\delta_1}} \qquad R_5 := T_2^{r_x} \cdot v^{-r_{\delta_2}}$$
$$Y := d_0^{r_{r_V}} \prod_{i=1}^{L} d_i^{r_{m_i}}$$

and sends $(T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5, Y)$ *to* V.

3. V *chooses* $c \leftarrow \mathbb{Z}_p$ *and sends it to* P.
4. P *computes and sends back to* V*:*

$$\begin{aligned} s_\alpha &= r_\alpha + c \cdot \alpha & s_\beta &= r_\beta + c \cdot \beta \\ s_x &= r_x + c \cdot x & s_{\delta_1} &= r_{\delta_1} + c \cdot \delta_1 \\ s_{\delta_2} &= r_{\delta_2} + c \cdot \delta_2 & s_{m_i} &= r_{m_i} + c \cdot m_i \text{ , for } i = 1, \ldots L \\ s_s &= r_s + c \cdot s & s_{r_V} &= r_{r_V} + c \cdot r_V \end{aligned}$$

5. V *verifies that the following holds:*

$$\begin{aligned} R_1 &= u^{s_\alpha} \cdot T_1^{-c} \\ R_2 &= v^{s_\beta} \cdot T_2^{-c} \\ R_3 &= e(T_3, g_2)^{s_x} \cdot e(z, w)^{-s_\alpha - s_\beta} \cdot e(z, g_2)^{-s_{\delta_1} - s_{\delta_2}} \cdot e(h_0, g_2)^{-s_s} \cdot \\ &\quad e(h_1, g_2)^{-s_{m_1}} \cdot \ldots \cdot e(h_L, g_2)^{-s_{m_L}} \cdot \left(\frac{e(T_3, w)}{e(g_1, g_2)}\right)^c \\ R_4 &= T_1^{s_x} \cdot u^{-s_{\delta_1}} \\ R_5 &= T_2^{s_x} \cdot v^{-s_{\delta_2}} \\ Y &= C_V^{-c} \cdot d_0^{s_{r_V}} \cdot \prod_{i=1}^{L} d_i^{s_{m_i}} \end{aligned}$$

6. *If all verification equations are satisfied, then* V *accepts.*

Theorem 5.10. *Protocol 5.9 is a* Σ*-protocol for proving knowledge of the values* $(m_1, \ldots, m_L, s, r_V, \alpha, \beta, x, \delta_1, \delta_2)$ *such that* $C_V = d_0^{r_V} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L} \wedge T_1 = u^\alpha \wedge T_2 = v^\beta \wedge \frac{e(T_3,w)}{e(g_1,g_2)} = e(T_3, g_2)^{-x} \cdot e(z, w)^{\alpha+\beta} \cdot e(z, g_2)^{\delta_1+\delta_2} \cdot e(h_0, g_2)^s \cdot e(h_1, g_2)^{m_1} \cdot \ldots \cdot e(h_L, g_2)^{m_L} \wedge T_1^x = u^{\delta_1} \wedge T_2^x = v^{\delta_2}$.

Proof. The theorem follows directly from Theorem 2.29 and Theorem 5.5. □

In other terms Protocol 5.9 is a Σ-protocol for proving knowledge of the values $(A, x, s, m_1, \ldots, m_L, r_V)$ such that $C_V = d_0^{r_V} \cdot d_1^{m_1} \cdot \ldots \cdot d_L^{m_L}$ and Verify $\big(pk_O, (m_1, \ldots, m_L), (A, x, s)\big) = 1$. This formulation state it clear that we will use the Protocol 5.9 for proving knowledge of a signature on a committed value, where the signature was generated using the same committed values as in the commitment C_V.

5.3.1 Efficiency

The above protocol is a combination of Protocol 2.25 and Protocol 5.4. We have analyzed the total communicated data of Protocol 5.4 in 5.2.1. Thus, we will only analyze and add up the additional communication data of the combination with Protocol 2.25. Additional elements in the transcripts of the above Protocol 5.9 are $Y \in \mathbb{G}_1$ and $s_{r_V} \in \mathbb{Z}_p$. Therefore, the total communicated data is $3748 + L \cdot 170$ bits or around $468 + L \cdot 22$ bytes. As mentioned in 4.2.2 and 5.2.1, we use the parameters suggested by Boneh et al. [8, 9]. Therefore, we use a 170-bit prime p, a group $\mathbb{G}_1$ where each group element can be represented by 171 bits and an embedding degree of 6.

6 An Efficient Basic Anonymous Credential System based on the q-SDH Assumption

Finally, we will present how the previously shown signature schemes and protocols can be used to design a basic anonymous credential system. We follow the Definition 3.5 for a basic anonymous credential system and list the necessary schemes and protocols. Since it is an instantiation of Definition 3.5, our anonymous credential system is secure and provides the basic properties, see Theorem 3.6. The basic properties are unforgeability of credentials, unlinkability of credential showings, consistency of credentials and the anonymity of users. For more details on the basic properties see 3.1.1. We call it efficient since the protocols have communication complexity linear in the security parameter and the signature scheme can be realized using groups with short representations. We analyzed the efficiency regarding the signature length and communicated data in 4.2.2, 5.1.1, 5.2.1 and 5.3.1. Let us first give a high-level description of our anonymous credential system and its functionality.

6.1 High-Level Description

In our basic anonymous credential system every organization O has its own instance of the signature scheme BBS-B (Definition 4.4), with corresponding signature public key pk_O and secret key sk_O. A commitment schemes public key is included in its public key pk_O. A user U has a secret, for example his name or a valid key of an public-key infrastructure. The Verifiers V use the common commitment schemes public key $pk_{\mathcal{C}}$ to verify any credential of a user U or to establish a pseudonym with a user U. Let us illustrate the system by an example. A user forms a pseudonym with an organization O by creating a commitment to his secret, using the commitment public key of the organization, that is included in pk_O. Consider a user U that wants to obtain a credential from organization O_1. Organization O_1 requires that a user possess credentials from O_2, O_3 and O_4. Assume user U already possesses credentials from O_2, O_3 and O_4. U gets a credential from O_1 by first establishing a pseudonym N_{O_1} with O_1. Then U proves to O_1 the knowledge of the credentials from O_2, O_3 and O_4. Now O_1 knows that the user, that it knows under N_{O_1}, possesses credentials from O_2, O_3 and O_4. O_1 will now grant U a credential for the pseudonym N_{O_1}.

Next we want to describe the structure of pseudonyms in an anonymous credential systems. A user U forms a pseudonym with an organization O, through forming a commitment to a secret m (e.g. his encoded identity) that is only known to U. A basic pseudonym is $C = h_0^s \cdot h_1^m$ where the element s is a random element and randomizes the resulting pseudonym and information-theoretically hides the secret. Therefore the user U can form several pseudonyms with the same secret m, but different random elements. Let us look at a more sophisticated anonymous credential system designed with a signature scheme for blocks of messages. Here, a pseudonym is formed using a commitment like $C = h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}$. Where only $m_1, \ldots m_k$ for $k < L$ corresponds to the user's secret. As before a random element s is used to randomize the pseudonym (respectively commitment). The other $L - k$ elements m_i can be used to encode attributes, for example expiration dates, user or system attributes. Lysyanskaya [34, Section 3.4] describes how to adapt the protocols to incorporate expiration dates. Next, we will list all protocols and schemes of our anonymous credential system. After that we will describe their use in the transactions between users, organizations and verifiers.

6.2 Basic Anonymous Credential System

Let us first give an overview of the building blocks that we have presented in this thesis and are now used in our anonymous credential system.

- A perfectly hiding and computationally binding Commitment Scheme $\mathcal{C}^L = (\mathrm{Gen}_{\mathcal{C}}, \mathrm{Commit})$ Definition 2.10
- A secure Signature Scheme $\Pi_B = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ Definition 4.4
- A protocol for proving knowledge of a committed value Protocol 2.25
- A protocol for signing a committed value Protocol 5.1
- A protocol for proving knowledge of a signature on a committed value Protocol 5.9

In the following we alter Gen and $\mathrm{Gen}_{\mathcal{C}}$ to exclude the run of the bilinear group generator $\mathcal{G}$. To be exact we let the *Init* algorithm set up the bilinear group pair using $\mathcal{G}$. To allow shorter notation in the following we assume that the output of $\mathcal{G}$ is known to everyone and therefore it is an implicit part of the public keys used in the transactions.

Definition 6.1. The basic anonymous credential system based on the q-SDH Assumption is defined as follows.

Init(1^η)

1. Gets $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$

2. Runs $\mathrm{Gen}_{\mathcal{C}}(1^\eta)$ and obtains $pk_{\mathcal{C}} = (u'_0, \ldots, u'_L) \in \mathbb{G}_2^{L+1}$ the common commitment public key

3. Outputs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e, pk_{\mathcal{C}}, L)$, where L has polynomial length in η

User Initialization User U chooses a secret $M_U \in \mathbb{Z}_p^L$, where $M_U = m_1, \ldots, m_L$.

Organization Initialization Organization O runs $\mathrm{Gen}(1^\eta)$ of the BBS-B signature scheme and gets a signature key pair $pk_O = (g_1, g_2, u_0, \ldots, u_L, w)$ and secret key $sk_O = \gamma_O$. It publishes pk_O as his public key.

FormNym(U, O) User U creates a pseudonym to his secret M_U by forming a commitment to M_U. Therefore, U chooses $s' \leftarrow \mathbb{Z}_p$ and computes $N_O = \mathrm{Commit}((u_0, \ldots, u_L), M_U, s')$. U sends N_O to organization O. Next, U runs with O the protocol for proving knowledge of a committed value with common input $(u_0, \ldots, u_L)$ and private input $(s', m_1, \ldots, m_L)$ to U. If O accepts the proof, then U and O store both N_O as U's pseudonym with O.

GrantCred(U, N_O, O) N_O is U's pseudonym established with O through a *FormNym*(U, O) transaction. Where $N_O = \mathrm{Commit}((u_0, \ldots, u_L), M_U, s')$ and $s' \leftarrow \mathbb{Z}_p$. User U and organization O run the protocol for signing a committed value on common input (pk_O, N_O). Where the user's private input is (s', M_U) and the organizations private input is sk_O. The user stores his output $\sigma = (A, x, s)$ of the protocol. Where σ is a valid signature on N_O under sk_O and called a *credential* in the following.

VerifyCred(U, V, N_O, O) User U has previously participated in a transaction *GrantCred*(U, N_O, O) and obtained the credential $\sigma = (A, x, s)$ from it. U forms a commitment $N_V = \mathrm{Commit}(pk_{\mathcal{C}}, M_U, r_V)$ to his secret M_U, where $r_V \leftarrow \mathbb{Z}_p$. User U and verifier V run the protocol for proving knowledge of a signature on a committed value. Where the common input to U and V is $(pk_O, pk_{\mathcal{C}}, N_V)$ and the private input to U is (A, x, s, r_V, M_U). V accepts the credential iff it accepts the proof.

VerifyCredOnNym(U, V, N_V, N_O, O) User U has previously participated in a transaction *GrantCred*(U, N_O, O) and obtained the credential $\sigma = (A, x, s)$ from it. Where $N_O = \mathrm{Commit}((u_0, \ldots, u_L), M_U, s')$ and $s' \leftarrow \mathbb{Z}_p$.
Further, U has established the pseudonym $N_V = \mathrm{Commit}(pk_{\mathcal{C}}, M_U, r_V)$ with V, where $r_V \leftarrow \mathbb{Z}_p$. User U and verifier V run the protocol for proving knowledge of a signature on a committed value. Where the common input to U and V is $(pk_O, pk_{\mathcal{C}}, N_V)$ and the private input for U is (A, x, s, r_V, M_U).

Notice that we have two types of pseudonyms in our system. First, the pseudonyms (resp. commitments) established with an organization. In this case a user forms the commitment, using the commitment public key of the corresponding organization. Second, there are pseudonyms established with verifiers. In this case

the commitment is formed using the common commitment public key $pk_{\mathcal{C}}$. The common commitment public key $pk_{\mathcal{C}}$ is initialized and outputted by the *Init* algorithm. We have introduced the two commitment public keys for efficiency reasons. Since our signature schemes directly provide a commitment public key included in its own signature public key.

6.3 Improvements

We want to design an efficient anonymous credential system. Therefore, we want to show some simple improvements that can be made to the presented signature schemes and protocols. The improvements give minor performance benefits. For example fewer uses of the bilinear map. In the case of the efficient protocol for proving knowledge of a signature, we talk about fewer bytes that are exchanged.

6.3.1 An Efficient Verify Algorithm for the Signature Schemes

Let us present a more efficient signature verification algorithm. It is suitable for our signature schemes (BBS-A Definition 4.1, BBS-B Definition 4.4) and requires fewer bilinear mappings. We will present it for the BBS-B signature scheme. The adaptions for the BBS-A signature scheme are minor.

Definition 6.2 (BBS-B with efficient Verify)**.** Let $\Pi_B = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ be the scheme of Definition 4.4 with Verify defined as follows:

Verify($\boldsymbol{pk}, \boldsymbol{M}, \boldsymbol{\sigma}$):

1. On input public key $pk = (g_1, g_2, u_0, u_1, \ldots, u_L, w,\ L,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ p,\ \psi,\ e)$, message block $M = (m_1, \ldots, m_L)$ and signature $\sigma = (A, x, s)$.
2. Set $h_i := \psi(u_i)$ for $i = 0, \ldots, L$
3. Output 1 if $e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) = e(g_1, g_2)$ holds. Otherwise output 0.

We replace it with the following algorithm.

VerifyEff($\boldsymbol{pk}, \boldsymbol{M}, \boldsymbol{\sigma}$):

1. On input public key $pk = (g_1, g_2, u_0, u_1, \ldots, u_L, w,\ L,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T,\ p,\ \psi,\ e)$, message block $M = (m_1, \ldots, m_L)$ and signature $\sigma = (A, x, s)$.
2. Set $h_i := \psi(u_i)$ for $i = 0, \ldots, L$
3. Output 1 if $e(A, wg_2^x) = e(g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}, g_2)$ holds. Otherwise output 0.

The check in step 3 of VerifyEff involves two computations of a bilinear mapping (respectively pairing). In comparison the old check presented above from Definition 4.4 involves four bilinear mappings. The check can also be used for our BBS-A signature scheme (Definition 4.1) with $L = 1$.

Lemma 6.3. *The* BBS-B *signature scheme of Definition 4.4 with* VerifyEff *is correct.*

Proof. Assume $\sigma \leftarrow \mathrm{Sign}(pk, sk, M)$ where $M = (m_1, \ldots, m_L)$, then $\sigma = (A, x, s)$ with $A = (g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L})^{\frac{1}{x+\gamma}}$ and $x, s, m_i \in \mathbb{Z}_p$ for $i = 1, \ldots, L$. Then $\mathrm{VerifyEff}(pk, m, \sigma)$ checks if the following holds.

$$\begin{aligned} e(A, wg_2^x) &= e(g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}, g_2) \Leftrightarrow \\ e((g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L})^{\frac{1}{x+\gamma}}, g_2^{x+\gamma}) &= e(g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}, g_2) \Leftrightarrow \\ e((g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}), g_2) &= e(g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L}, g_2) \end{aligned}$$

□

Next, we will show an efficient protocol that substitute the Protocol 5.4, for proving knowledge of a signature of the signature scheme BBS-B.

6.3.2 An Efficient Protocol for Proving Knowledge of a Signature

The following protocol is a variant of the protocol presented in [8, 10]. A similar protocol is presented by Au et al. [2] in another context. The protocol presented in this section proves the possession of a valid signature on a block of messages from the BBS-B signature scheme (Definition 4.4). The protocol is also applicable for the BBS-A signature scheme of Definition 4.1.

Protocol 6.4. *Let $\mathcal{G}$ be a bilinear group generator and $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e) \leftarrow \mathcal{G}(1^\eta)$. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$ such that $g_1 = \psi(g_2)$. Further, let $\Pi_B = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Verify})$ be the secure signature scheme* BBS-B *of Definition 4.4 and $(pk, sk) \leftarrow \mathrm{Gen}(1^\eta)$, where $pk := (g_1, g_2, u_0, \ldots, u_L, w)$. The protocol proves the possession of the elements $\sigma = (A, x, s) \in \mathbb{G}_1 \times \mathbb{Z}_p^2$ and $m_1, \ldots, m_L \in \mathbb{Z}_p^L$ such that $\mathrm{Verify}(pk, (m_1, \ldots m_L), \sigma) = 1$. The value A is uniquely determined by $A = (g_1 \cdot h_0^s \cdot h_1^{m_1} \cdot \ldots \cdot h_L^{m_L})^{\frac{1}{x+\gamma}}$, where the value $\gamma \in \mathbb{Z}_p$ is secret and not known to the prover. From $\mathrm{Verify}(pk, (m_1, \ldots m_L), \sigma) = 1$ we know that the verification equation $e(A, g_2)^x \cdot e(A, w) \cdot e(h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2) = e(g_1, g_2)$ is satisfied. The protocol for proving knowledge of a signature on a block of messages is defined as follows.*

Common Input *The signature public key $pk = (g_1, g_2, u_0, \ldots, u_L, w)$ and $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \psi, e)$. Additional public values are $g_1, h_0, \ldots, h_L, u, v, z \in \mathbb{G}_1$, $g_2, w \in \mathbb{G}_2$ and L, where L has polynomial length in η, and $g_2, u_0, \ldots, u_L \leftarrow \mathbb{G}_2$, $u, v, z \leftarrow \mathbb{G}_1$, $h_i = \psi(u_i)$ for $i = 0, \ldots, L$, $w = g_2^\gamma$ and $g_1 = \psi(g_2)$*

User's private Input *Values $A \in \mathbb{G}_1$ and $x, s, m_1, \ldots, m_L \in \mathbb{Z}_p$ where $\sigma = (A, x, s)$ such that*

$$\text{Verify}(pk, (m_1, \ldots, m_L), \sigma) = 1$$

Protocol

1. *Prover P selects exponents $\alpha, \beta \leftarrow \mathbb{Z}_p$ and computes:*

$$T_1 = u^\alpha \cdot v^\beta \qquad\qquad T_2 = A \cdot z^\alpha$$

   P also computes two helper values:

$$\delta_1 := x\alpha \qquad\qquad \delta_2 := x\beta$$

   P and V then undertake a proof of knowledge of values (α, β, x, s, m_1, $\ldots, m_L$, δ_1, δ_2).

2. *P chooses blinding values $r_\alpha, r_\beta, r_x, r_s, r_{m_1}, \ldots, r_{m_L}, r_{\delta_1}, r_{\delta_2} \leftarrow \mathbb{Z}_p$ and computes:*

$$R_1 := u^{r_\alpha} \cdot v^{r_\beta}$$

$$R_2 := e(T_2, g_2)^{r_x} \cdot e(z, w)^{-r_\alpha} \cdot e(z, g_2)^{-r_{\delta_1}} \cdot e(h_0, g_2)^{-r_s} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-r_{m_i}}$$

$$R_3 := T_1^{r_x} \cdot u^{-r_{\delta_1}} \cdot v^{-r_{\delta_2}}$$

   and sends $(T_1, T_2, R_1, R_2, R_3)$ to V.

3. *V chooses $c \leftarrow \mathbb{Z}_p$ and sends it to P.*

4. *P computes and sends back to V:*

$$\begin{aligned} s_\alpha &= r_\alpha + c \cdot \alpha & s_\beta &= r_\beta + c \cdot \beta \\ s_x &= r_x + c \cdot x & s_{\delta_1} &= r_{\delta_1} + c \cdot \delta_1 \\ s_{\delta_2} &= r_{\delta_2} + c \cdot \delta_2 & s_{m_i} &= r_{m_i} + c \cdot m_i \text{ , for } i = 1, \ldots L \\ s_s &= r_s + c \cdot s \end{aligned}$$

5. *V verifies that the following holds:*

$$u^{s_\alpha} \cdot v^{s_\beta} \cdot T_1^{-c} = R_1 \tag{6.1}$$

$$e(T_2, g_2)^{s_x} \cdot e(z, w)^{-s_\alpha} \cdot e(z, g_2)^{-s_{\delta_1}} \cdot e(h_0, g_2)^{-s_s} \cdot e(h_1, g_2)^{-s_{m_1}} \cdot \ldots \cdot e(h_L, g_2)^{-s_{m_L}} \cdot \left(\frac{e(T_2, w)}{e(g_1, g_2)}\right)^c = R_2 \tag{6.2}$$

$$T_1^{s_x} \cdot u^{-s_{\delta_1}} \cdot v^{-s_{\delta_2}} = R_3 \tag{6.3}$$

6. *If all three verification equations are satisfied, then V accepts.*

Theorem 6.5. *Protocol 6.4 is a Σ-protocol for proving knowledge of the values $(\alpha, \beta, x, s, m_1, \ldots, m_L, \delta_1, \delta_2)$ such that $T_1 = u^\alpha \cdot v^\beta \wedge \frac{e(T_2,w)}{e(g_1,g_2)} = e(T_2, g_2)^{-x} \cdot e(z, w)^\alpha \cdot e(z, g_2)^{\delta_1} \cdot e(h_0, g_2)^s \cdot e(h_1, g_2)^{m_1} \cdot \ldots \cdot e(h_L, g_2)^{m_L} \wedge T_1^x = u^{\delta_1} \cdot v^{\delta_2}$.*

Proof. The theorem follows from Lemma 6.6, Lemma 6.7 and Lemma 6.8. □

First we show that the protocol is complete; second, the protocol is a special honest-verifier zero-knowledge protocol; and third, the protocol satisfies the special soundness property by giving an appropriate extractor.

Lemma 6.6. *Protocol 6.4 is complete.*

Proof. Assume a honest prover P in possession of $(A, x, s, m_1, \ldots, m_L)$, such that $A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L} = g_1$. P then follows the specified steps of the protocol. Hence, for the verification equation for R_2 we get the following.

$$
\begin{aligned}
& e(T_2, g_2)^{s_x} \cdot e(z, w)^{-s_\alpha} \cdot e(z, g_2)^{-s_{\delta_1}} \cdot e(h_0, g_2)^{-s_s} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-s_{m_i}} \cdot \left(\frac{e(T_2, w)}{e(g_1, g_2)}\right)^c \\
&= \frac{e(T_2, g_2)^{cx} \cdot e(z, w)^{-c\alpha} \cdot e(z, g_2)^{-c\delta_1} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_2}{e(T_2, w)^{-c} e(g_1, g_2)^c} \\
&= \frac{e(Az^\alpha, g_2)^{cx} \cdot e(z, w)^{-c\alpha} \cdot e(z, g_2)^{cx\cdot(-\alpha)} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_2}{e(T_2, w)^{-c} e(g_1, g_2)^c} \\
&= \frac{e(Az^\alpha, g_2)^{cx} \cdot e(z, w)^{-c\alpha} \cdot e(z^{-\alpha}, g_2)^{cx} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_2}{e(T_2, w)^{-c} e(g_1, g_2)^c} \\
&= e(A, g_2)^{cx} e(z, w)^{-c\alpha} \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_2 \cdot \left(\frac{e(T_2, w)}{e(g_1, g_2)}\right)^c \\
&= e(A, g_2)^{cx} e(z^{-\alpha}, w)^c \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_2 \cdot \left(\frac{e(Az^\alpha, w)}{e(g_1, g_2)}\right)^c \\
&= e(A, g_2)^{cx} e(A, w)^c \cdot e(h_0, g_2)^{-cs} \cdot \prod_{i=1}^{L} e(h_i, g_2)^{-cm_i} \cdot R_2 \cdot \left(\frac{1}{e(g_1, g_2)}\right)^c \\
&= e(A^x, g_2)^c e(A^\gamma, g_2)^c \cdot e(h_0^{-s}, g_2)^c \cdot \prod_{i=1}^{L} e(h_i^{-m_i}, g_2)^c \cdot R_2 \cdot \left(\frac{1}{e(g_1, g_2)}\right)^c \\
&= R_2 \cdot \left(\frac{e(A^{x+\gamma} \cdot h_0^{-s} \cdot h_1^{-m_1} \cdot \ldots \cdot h_L^{-m_L}, g_2)}{e(g_1, g_2)}\right)^c \\
&= R_2 \cdot \left(\frac{e(g_1, g_2)}{e(g_1, g_2)}\right)^c \\
&= R_2
\end{aligned}
$$

Regrading the verification equations for R_1 and R_3 the following holds.

$$u^{s_\alpha} \cdot v^{s_\beta} \cdot T_1^{-c} = u^{r_\alpha + c\alpha} \cdot v^{r_\beta + c\beta} \cdot u^{-c\alpha} \cdot v^{-c\beta} = u^{r_\alpha} \cdot v^{r_\beta} = R_1$$

$$\begin{aligned} T_1^{s_x} \cdot u^{-s_{\delta_1}} \cdot v^{-s_{\delta_2}} &= u^{\alpha(r_x + cx)} \cdot u^{-r_{\delta_1} - \alpha cx} \cdot v^{\beta(r_x + cx)} \cdot v^{-r_{\delta_1} - \beta cx} \\ &= T_1^{r_x} \cdot u^{-r_{\delta_1}} \cdot v^{-r_{\delta_2}} \\ &= R_3 \end{aligned}$$

All verification equations are satisfied. Therefore, verifier V will always accept if prover P is honest. □

Lemma 6.7. *Protocol 6.4 is a (perfect) special honest-verifier zero-knowledge protocol under the assumption that the discrete logarithm problem is hard in $\mathbb{G}_1$.*

Proof. We describe a simulator $\mathcal{S}$ which outputs transcripts of Protocol 6.4. $\mathcal{S}$ on input the public parameters and challenge $c' \in \mathbb{Z}_p$, chooses $\alpha', \beta' \leftarrow \mathbb{Z}_p$, $A' \leftarrow \mathbb{G}_1$ and sets:

$$T_1 := u^{\alpha'} \cdot v^{\beta'} \qquad\qquad T_2 := A' \cdot z^{\alpha'}$$

$\mathcal{S}$ picks $s_{\alpha'}, s_{\beta'}, s_{x'}, s_{s'}, s_{m'_1}, \ldots, s_{m'_L}, s_{\delta'_1}, s_{\delta'_2} \leftarrow \mathbb{Z}_p$ and sets

$$\begin{aligned} R_1 &= u^{s_{\alpha'}} \cdot v^{s_{\beta'}} \cdot T_1^{-c'} \\ R_2 &= e(T_2, g_2)^{s_{x'}} \cdot e(z, w)^{-s_{\alpha'}} \cdot e(z, g_2)^{-s_{\delta'_1}} \cdot e(h_0, g_2)^{-s'} \cdot \\ &\quad e(h_1, g_2)^{-s_{m'_1}} \cdot \ldots \cdot e(h_L, g_2)^{-s_{m'_L}} \cdot \left(\frac{e(T_2, w)}{e(g_1, g_2)}\right)^{c'} \\ R_3 &= T_1^{s_{x'}} \cdot u^{-s_{\delta'_1}} \cdot v^{-s_{\delta'_2}} \end{aligned}$$

Next, $\mathcal{S}$ outputs the transcript (T_1, T_2, R_1, R_2, R_3, c', $s_{\alpha'}$, $s_{\beta'}$, $s_{x'}$, $s_{s'}$, $s_{m'_1}, \ldots, s_{m'_L}$, $s_{\delta'_1}$, $s_{\delta'_2}$). The verification equations are directly satisfied, because $\mathcal{S}$ sets R_1, R_2 and R_3 accordingly. In detail, $\mathcal{S}$ chooses the values $s_{\alpha'}, s_{\beta'}, s_{x'}, s_{s'}, s_{m'_1}, \ldots, s_{m'_L}$, $s_{\delta'_1}$ and $s_{\delta'_2}$ uniformly at random from $\mathbb{Z}_p$. After that $R_1, R_2, R_3, R_4, R_5, R_6, R_7$ are set, such that the verification equations are satisfied. Let us now analyze the probability of the transcript that $\mathcal{S}$ given challenge c' produces. The values T_1 and T_2 are independent from the challenge and only determined by the choice of α',β' and A'. Where the value A' is as in the real protocol a random element in $\mathbb{G}_1$. Further, as in the real protocol α' and β' are chosen uniformly at random from $\mathbb{Z}_p$. The $L+6$ values $s_{\alpha'}, s_{\beta'}, s_{x'}, s_{s'}, s_{m'_1}, \ldots, s_{m'_L}, s_{\delta'_1}, s_{\delta'_2}$ are also chosen uniformly at random from $\mathbb{Z}_p$. Hence, the probability of $\mathcal{S}$ to output the transcript given challenge c' is $\frac{1}{p^2} \cdot \frac{1}{p^{L+6}} = \frac{1}{p^{L+8}}$. The same holds for the transcripts in the real protocol conditioned on $c = c'$. There we have that $s_\alpha, s_\beta, s_x, s_{\delta_1}$ and s_{δ_2} are fixed after α, β, r_α, r_β, r_x, $r_{s'}$, r_{m_1}, $\ldots, r_{m_L}$, r_{δ_1}, r_{δ_2} were chosen uniformly at random from $\mathbb{Z}_p$. It follows that the probability distribution of the transcripts of the real protocol and S given challenge $c = c'$ are equal. □

Lemma 6.8. *Protocol 6.4 satisfies the special soundness property.*

Proof. We show that there is an extractor $\mathcal{E}$ that can extract a witness given two accepting transcripts of Protocol 6.4. Assume $\mathcal{E}$ is given the two accepting transcripts $(T_1, T_2, R_1, R_2, R_3, c, s_\alpha, s_\beta, s_x, s_s, s_{m_1}, \ldots, s_{m_L}, s_{\delta_1}, s_{\delta_2})$ with challenge c and $(T_1, T_2, R_1, R_2, R_3, c', s'_\alpha, s'_\beta, s'_x, s'_s, s'_{m_1}, \ldots, s'_{m_L}, s'_{\delta_1}, s'_{\delta_2})$ with challenge c' where $c \neq c'$. $\mathcal{E}$ extracts the witness as follows. It first sets the following helper values:

$$
\begin{aligned}
\Delta c &:= c - c' & \Delta s_\alpha &:= s_\alpha - s'_\alpha & \Delta s_{m_i} &:= s_{m_i} - s'_{m_i}\text{ , for } i = 1, \ldots, L \\
\Delta s_\beta &:= s_\beta - s'_\beta & \Delta s_x &:= s_x - s'_x & \Delta s_{\delta_1} &:= s_\delta - s'_\delta \\
\Delta s_{\delta_2} &:= s_{\delta_2} - s'_{\delta_2} & \Delta s_s &:= s_s - s'_s \\
\tilde{\alpha} &:= \frac{\Delta s_\alpha}{\Delta c} & \tilde{\beta} &:= \frac{\Delta s_\beta}{\Delta c} & \tilde{m_i} &:= \frac{\Delta s_{m_i}}{\Delta c}\text{ , for } i = 1, \ldots, L \\
\tilde{x} &:= \frac{\Delta s_x}{\Delta c} & \tilde{s} &:= \frac{\Delta s_s}{\Delta c}
\end{aligned}
$$

Following the verification equations and using both transcripts we get the following.

$$
\begin{aligned}
u^{s_\alpha} \cdot v^{s_\beta} \cdot T_1^{-c} \cdot u^{-s'_\alpha} \cdot v^{-s'_\beta} \cdot T_1^{c'} = u^{s_\alpha - s'_\alpha} \cdot v^{s_\beta - s'_\beta} \cdot T_1^{c'-c} &= u^{\Delta s_\alpha} \cdot v^{\Delta s_\beta} \cdot T_1^{-\Delta c} \\
&\Rightarrow T_1^{\Delta c} = u^{\Delta s_\alpha} \cdot v^{\Delta s_\beta} \Rightarrow T_1 = u^{\tilde{\alpha}} \cdot v^{\tilde{\beta}}
\end{aligned}
$$

$$
\begin{aligned}
T_1^{s_x} \cdot u^{-s_{\delta_1}} \cdot v^{-s_{\delta_2}} \cdot T_1^{-s'_x} \cdot u^{s'_{\delta_1}} \cdot v^{s'_{\delta_2}} &= T_1^{s_x - s'_x} \cdot u^{s'_{\delta_1} - s_{\delta_1}} \cdot v^{s'_{\delta_2} - s_{\delta_2}} \\
&= T_1^{\Delta s_x} \cdot u^{-\Delta s_{\delta_1}} \cdot v^{-\Delta s_{\delta_2}} \\
&\Rightarrow T_1^{\Delta s_x} = u^{\Delta s_{\delta_1}} \cdot v^{\Delta s_{\delta_2}} \\
&\Rightarrow \tilde{\alpha} \Delta s_x + \tilde{\beta} \Delta s_x - \Delta s_{\delta_2} = \Delta s_{\delta_1}
\end{aligned}
$$

Observe that the following holds and is later used in Eq. (6.4).

$$
\begin{aligned}
\Delta s_{\delta_1} &= \tilde{\alpha} \Delta s_x + \tilde{\beta} \Delta s_x - \Delta s_{\delta_2} \\
&= \tilde{\alpha} \Delta s_x + \tilde{\beta} \Delta s_x - s_{\delta_2} + s'_{\delta_2} \\
&= \tilde{\alpha} \Delta s_x + \tilde{\beta} \Delta s_x - c\delta_2 + c'\delta_2 \\
&= \tilde{\alpha} \Delta s_x + \tilde{\beta} \Delta s_x - \beta x (c - c')
\end{aligned}
$$

Taking the Δc -th root and with $\tilde{x} = x$, $\tilde{\beta} = \beta$ we obtain

$$
\begin{aligned}
\frac{\Delta s_{\delta_1}}{\Delta c} &= \frac{\tilde{\alpha} \Delta s_x}{\Delta c} + \frac{\tilde{\beta} \Delta s_x}{\Delta c} - \frac{\beta x (c - c')}{\Delta c} \\
&= \tilde{\alpha} \tilde{x} + \tilde{\beta} \tilde{x} - \beta x \\
&= \tilde{\alpha} \tilde{x}
\end{aligned}
$$

Dividing the two instances of the verification equation for R_2 we get,

$$\frac{e(T_2,g_2)^{s_x}\cdot e(z,w)^{-s_\alpha}\cdot e(z,g_2)^{-s_{\delta_1}}\cdot e(h_0,g_2)^{-s_s}\cdot\prod_{i=1}^{L}e(h_i,g_2)^{-s_{m_i}}\cdot\left(\frac{e(T_2,w)}{e(g_1,g_2)}\right)^{c}}{e(T_2,g_2)^{s'_x}\cdot e(z,w)^{-s'_\alpha}\cdot e(z,g_2)^{-s'_{\delta_1}}\cdot e(h_0,g_2)^{-s'_s}\cdot\prod_{i=1}^{L}e(h_i,g_2)^{-s'_{m_i}}\cdot\left(\frac{e(T_2,w)}{e(g_1,g_2)}\right)^{c'}}$$
$$=e(T_2,g_2)^{\Delta s_x}\cdot e(z,w)^{-\Delta s_\alpha}\cdot e(z,g_2)^{-\Delta s_{\delta_1}}\cdot e(h_0,g_2)^{-\Delta s_s}\cdot$$
$$e(h_1,g_2)^{-\Delta s_{m_1}}\cdot\ldots\cdot e(h_L,g_2)^{-\Delta s_{m_L}}\cdot\left(\frac{e(T_2,w)}{e(g_1,g_2)}\right)^{\Delta c}$$

Which is equivalent to

$$\begin{aligned}\left(\frac{e(g_1,g_2)}{e(T_2,w)}\right)^{\Delta c} =& e(T_2,g_2)^{\Delta s_x}\cdot e(z,w)^{-\Delta s_\alpha}\cdot e(z,g_2)^{-\Delta s_{\delta_1}}\cdot e(h_0,g_2)^{-\Delta s_s}\cdot\\ & e(h_1,g_2)^{-\Delta s_{m_1}}\cdot\ldots\cdot e(h_L,g_2)^{-\Delta s_{m_L}} \qquad \Leftrightarrow\\ \left(\frac{e(g_1,g_2)}{e(T_2,w)}\right) =& e(T_2,g_2)^{\tilde{x}}\cdot e(z,w)^{-\tilde{\alpha}}\cdot e(z,g_2)^{-\tilde{x}\tilde{\alpha}}\cdot e(h_0,g_2)^{-\tilde{s}}\cdot\\ & e(h_1,g_2)^{-\tilde{m}_1}\cdot\ldots\cdot e(h_L,g_2)^{-\tilde{m}_L} \qquad \Leftrightarrow\\ e(g_1,g_2) =& e(T_2,g_2)^{\tilde{x}}\cdot e(z^{-\tilde{\alpha}},w)\cdot e(z^{-\tilde{\alpha}},g_2)^{\tilde{x}}\cdot e(h_0,g_2)^{-\tilde{s}}\cdot\\ & e(h_1,g_2)^{-\tilde{m}_1}\cdot\ldots\cdot e(h_L,g_2)^{-\tilde{m}_L}\cdot e(T_2,w) \qquad \Leftrightarrow\\ e(g_1,g_2) =& e(T_2z^{-\tilde{\alpha}},g_2)^{\tilde{x}}\cdot e(T_2z^{-\tilde{\alpha}},w)\cdot e(h_0,g_2)^{-\tilde{s}}\cdot\prod_{i=1}^{L}e(h_i,g_2)^{-\tilde{m}_i}\end{aligned} \tag{6.4}$$

with $\tilde{A}=T_2z^{-\tilde{\alpha}}$ we get:

$$e(g_1,g_2) = e(\tilde{A},g_2)^{\tilde{x}}\cdot e(\tilde{A},w)\cdot e(h_0,g_2)^{-\tilde{s}}\cdot e(h_1,g_2)^{-\tilde{m}_1}\cdot\ldots\cdot e(h_L,g_2)^{-\tilde{m}_L}$$

The result $(\tilde{A},\tilde{x},\tilde{s},\tilde{m}_1,\ldots,\tilde{m}_L)$ is a valid signature $\tilde{\sigma}=(\tilde{A},\tilde{x},\tilde{s})$ on the block of messages $\tilde{m}_1,\ldots,\tilde{m}_L$. □

Notice that the above Protocol 6.4 is proven to be a perfect special honest-verifier zero-knowledge protocol. Whereas, Protocol 5.4 is only computational special honest-verifier zero-knowledge under the assumption that the Decision Linear Problem holds in $\mathbb{G}_1$.

Next, we want to analyze the communicated data of Protocol 6.4. We follow the parameters of Boneh et al. [9, 8] and take p to be a 170-bit prime and use a group $\mathbb{G}_1$, where each group element can be represented by 171 bits. The total communicated data in our Protocol 6.4 with embedding degree 6 is $2894+L\cdot 170$ bits or around $362+L\cdot 22$ bytes. Thus, for our BBS-A signature scheme with $L=1$ it results in about 383 bytes. Let us compare it to Protocol 5.4, which is an extension of the protocol originally presented by Boneh et al. [8]. We analyzed in 5.2.1, that the total communicated data of Protocol 5.4 is $3407+L\cdot 170$ bits. Hence, the overall savings are 513 bits. This is related to the three elements of

$\mathbb{G}_1$ lesser in the transcripts of the above Protocol 6.4 than in Protocol 5.4.

Considering long-term security and following Barreto et al. [3] one should take an embedding degree of 12. For Protocol 6.4 this results in $3744 + L \cdot 170$ bits or around $468 + L \cdot 22$ bytes of communication data.

Considering the usage of Protocol 6.4 in our anonymous credentials systems we have to combine it with the protocol for proving knowledge of a committed value Protocol 2.25. Thus we obtain a more efficient protocol for proving knowledge of signature on a committed value.

6.4 Additions and Open Questions

Desirable extensions, to our basic anonymous credential system, are the ability to handle misuse of credentials and to prevent users from sharing their secrets. We already presented this additional desirable properties in 3.1.2. The direct solution to handle a misuse of credentials, is to revoke them. Lysyanskaya [34] state that we do not know how to achieve this from the assumptions of signature and commitment schemes. One possible approach is to rely on dynamic accumulators as presented in [34].

Of interest is also a process to reveal a user's pseudonym with an issuing organization O, in the case that the user misuses his credential from O. Therefore, we want to discuss that our presented signature scheme BBS-B Definition 4.4 provides the same weaker form of anonymity revocation as mentioned in [34, Section 3.4.1]. This form of anonymity revocation is also called *local* anonymity revocation. Notice that the signer (resp. organization) O, in the protocol for signing a committed value (Protocol 5.1), outputs $(A, x, s'') \in \mathbb{G}_1 \times \mathbb{Z}_p^2$ to the user U. Let us denote the output by $s(\sigma)$. We alternate the $\mathrm{GrantCred}(U, N_O, O)$ transaction such that O also stores the tuple $(N_O, s(\sigma))$. Suppose the user proves the possession of a credential from O, for example with a $\mathrm{VerifyCred}(U, V, N_O, O)$ transaction. Then the user encrypts the value $s(\sigma)$ under some trusted party's public key. The proof is extended such that, the user proves that the encrypted value corresponds to his credential with O. In case of a misuse of the credential, the trusted party decrypts the ciphertext and obtains $s(\sigma)$. The value is securely communicated to every organization. The organizations check if they have stored tuple containing the value $s(\sigma)$. Therefore, the issuing organization O is able to link the value $s(\sigma)$ to the pseudonym N_O, which was used to grant the credential in question. Organization O can now take further actions, like publicly reveal the pseudonym N_O.

As mentioned before our signature schemes and protocols are efficient in regard to the signature length and total communicated data in the protocols. Our and other signature schemes that are secure under the q-SDH Assumption, like the schemes by Boneh et al. [6, 8], are presented in the so called Type 2 pairing setting. We will not describe the setting in detail, but one property of the Type 2 setting is that there is a efficiently computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$.

The signature scheme by Boneh et al. [6] was later reproduced for the so called Type 3 pairing setting in [7]. Without going into detail, for the Type 3 setting there is no efficiently computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ known. Following Chatterjee and Menezes [20], the Type 3 setting gives us on the other side smaller bit representation of elements in $\mathbb{G}_2$ and better performance for exponentiations in the groups. We motivate that our presented signature schemes and anonymous credential system can also be reproduced in the Type 3 setting.

Bibliography

[1] Giuseppe Ateniese and Breno de Medeiros. Efficient group signatures without trapdoors. In *Advances in Cryptology ASIACRYPT 2003*, pages 246–268. Springer, 2003.

[2] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-taa. In *Security and Cryptography for Networks*, pages 111–125. Springer, 2006.

[3] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected areas in cryptography*, pages 319–331. Springer, 2006.

[4] Amos Beimel, Tal Malkin, and Silvio Micali. The all-or-nothing nature of two-party secure computation. In *Advances in Cryptology-CRYPTO'99*, pages 80–97. Springer, 1999.

[5] Johannes Blömer, Jakob Juhnke, and Christina Kolb. Anonymous and publicly linkable reputation systems. In *Financial Cryptography and Data Security*, pages 478–488. Springer, 2015. To appear. Full version available at: `https://eprint.iacr.org/2014/546.pdf`.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology-EUROCRYPT 2004*, pages 56–73. Springer, 2004.

[7] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[8] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. *Advances in Cryptology-CRYPTO'04*, pages 41–55, 2004.

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology-ASIACRYPT 2001*, pages 514–532. Springer, 2001.

[10] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 168–177. ACM, 2004.

[11] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In *Advances in Cryptology-EUROCRYPT'97*, pages 318–333. Springer, 1997.

[12] Stefan A Brands. *Rethinking public key infrastructures and digital certificates: building in privacy.* MIT Press, 2000.

[13] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology—EUROCRYPT 2001*, pages 93–118. Springer, 2001.

[14] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in communication networks*, pages 268–289. Springer, 2003.

[15] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. *Advances in Cryptology-CRYPTO'04*, 2005.

[16] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology—CRYPTO'97*, pages 410–424. Springer, 1997.

[17] Ran Canetti. *Studies in Secure Multiparty Computation and Applications.* PhD thesis, The Weizmann Institute of Science, 1996.

[18] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[19] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.

[20] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings—the role of ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.

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

[22] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Advances in Cryptology-CRYPTO'86*, pages 118–167. Springer, 1987.

[23] Lidong Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms*, pages 232–243. Springer, 1996.

[24] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Advances in Cryptology-EUROCRYPT 2000*, pages 418–430. Springer, 2000.

[25] Ivan Damgård. On σ-protocols. *Lecture notes for CPT*, 2002.

[26] Ivan Bjerre Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *Proceedings on Advances in cryptology*, pages 328–335. Springer-Verlag New York, Inc., 1990.

[27] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976.

[28] Cynthia Dwork, Jeffrey Lotspiech, and Moni Naor. Digital signets: Self-enforcing protection of digital information. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 489–498. ACM, 1996.

[29] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology-CRYPTO'86*, pages 186–194. Springer, 1987.

[30] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*, volume 1 of *Foundations of Cryptography*. Cambridge University Press, 2001.

[31] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*, volume 2 of *Foundations of Cryptography*. Cambridge University Press, 2009.

[32] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[33] Marc Joye and Gregory Neven. *Identity-based cryptography*, volume 2. IOS press, 2009.

[34] Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2002.

[35] Anna Lysyanskaya, Ronald L Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199. Springer, 2000.

[36] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43. ACM, 1989.

[37] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology—CRYPTO'92*, pages 31–53. Springer, 1993.

[38] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology-CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, Springer Berlin Heidelberg, 1992.

[39] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 245–254. ACM, 2000.

[40] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[41] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394. ACM, 1990.

[42] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[43] Eric R Verheul. Self-blindable credential certificates from the weil pairing. In *Advances in cryptology ASIACRYPT 2001*, pages 533–551. Springer, 2001.