An Efficient Verification of CL-LRSW Signatures and a Pseudonym Certificate System

Marcin Słowik
Wrocław University of Science and Technology
marcin.slowik@pwr.edu.pl

Marta Wszoła
Wrocław University of Science and Technology
marta.wszola@pwr.edu.pl

ABSTRACT

We present an efficient interactive verification of Camenisch's and Lysyanskaya's LRSW-based signatures with hidden attributes. In our construction, a signature owner computes no bilinear mappings and does not operate in bilinear mapping function image group. Computation complexity is also reduced for a Verifier. We construct a pairing-based Pseudonymous Domain Certificate system which can be used as a building block for pseudonymous signatures and identification systems.

Keywords

Anonymous Credentials; Pseudonymous Signatures; Domain Signatures; Restricted Identification; Zero-Knowledge; Sigma Protocols; Pairing-Based Cryptography

1. INTRODUCTION

Nowadays, we are surrounded by organizations aggressively collecting personal data. Each time an average person surfs the internet, she shares plenty of her private information, such as her browsing history or even an accurate geographical location, with advertising third parties. Thus, more and more people search for a way to exchange small pieces of their personal data, without compromising their overall anonymity. Anonymous credentials systems has been in use for a while now. They provide the methods of sharing some pieces of information with other parties (traditionally called organizations) in such a way that system Users are not identified. The organizations learn only the fact that the Users are genuine "owners" of the information. Anonymous credential systems are used in various fields such as personal authentication (Identity Mixer [10], U-Prove [14]) and Direct Anonymous Attestation [7, 8]. Of course, in some cases it is useful to be remembered within a single organization but to be anonymous at the same time. Domain pseudonyms are IDs which cannot be linked, yet are related to User's original identity in such a way that she cannot use two different pseudonyms for the same domain [12]. Thus, the organization can track history of a single User but cannot identify her outside the organization. The domain pseudonymous signatures are deployed in German identity cards [2].

One of the goals in the anonymous credential system research is to design a scheme which requires as little memory space and computation power as possible. While there are widely deployed solutions, they utilize large RSA numbers and thus are not lightweight [3, 6]. The LRSW-based credential system introduced by Camenisch and Lysyanskaya in [4] provides credentials composed of elliptic curve points. However, the naive and straightforward implementation of the credential possession proof requires computation of bilinear mappings by both a Prover and a Verifier. While at a level of security equal to the security of 128-bit AES, computation of Tate pairings is less demanding than an exponentiation of RSA numbers [11], it may be problematic with small devices such as smartcards. Moreover, as of today, cryptographic libraries on such devices do not implement bilinear pairings at all.

Since credential presentation should be as fast as possible, it is only natural to replace interactive verification protocol with non-interactive certificates. Given the system with domains, the certificate may be computed once for each domain and a credential set and then reused.

In this paper, we first present a modification of Camenisch and Lysyanskaya's scheme in which the Prover has to perform only standard elliptic curve operations during credential verification process. We then provide related security proofs. Finally, we present, how to utilize the modified scheme in order to obtain Domain-Specific certificates for Pseudonymous Credentials and Pseudonymous Identification.

1.1 Overview

In Section 2, we give general definitions for the cryptographic and mathematical constructs we use. Section 2.1 describes a credential system on an abstract level, defining its entities, functions and basic properties. In Section 2.2 we give the formal definition of the zero-knowledge proofs of knowledge. In Section 2.3 we briefly describe the mathematical concept of bilinear maps which properties are essential for our construction. In Section 2.4 we recall Camenisch's and Lysyanskaya's signature scheme based on bilinear maps properties. We do not provide related security proofs as they can be found in their paper. Our construction is strongly based on this particular scheme. In Section 2.5 we recall Fiat-Shamir transformation changing

APKC'17, April 02 2017, Abu Dhabi, United Arab Emirates

DOI: http://dx.doi.org/10.1145/3055504.3055506

interactive zero-knowledge proofs into non-interactive signature schemes. We use it later for certificates construction.

In Section 3.1 we present a verification procedure for CL-LRSW signatures C, consisting of a proof that does not require any computation of bilinear mappings or operations in the image group of bilinear mappings on the side of the Prover, and greatly reduces the computational burden on Verifier as well. The change that we make in original scheme invalidates its security proof. The original credential verification protocol requires performing a certain zero-knowledge proof of knowledge and exponents equation as a separate building block. Since we do not perform it straightforwardly, we provide a new security proof for the zero-knowledge properties that remain unaffected after our modification.

In Section 3.2 we show a way to perform zero-knowledge proofs of knowledge of equality of any number of (sometimes entangled) exponents on different bases. Our change is focused on reducing the number of operations and intermediate values produced during protocol execution. We give a proof that our protocol is indeed a zero-knowledge proof of knowledge.

We show how to obtain an anonymous credential system from our constructs in Section 4 by defining system entities and functions utilizing modified verification protocol. The security properties of the system are induced by the properties of the original Camenisch's and Lysyanskaya's scheme as well as our security proofs.

In Section 5 we build a domain credential system. We apply Fiat-Shamir Transform on the proofs to create reusable certificates for scenarios, where Issuers are supposed to link different presentations of the same User, but should not be able to link them across different domains.

2. PRELIMINARIES

2.1 Credential System

The credential system consists of three types of entities, depending on their role:

1. Users – system Users possessing a set of qualities and attributes
2. Issuers – authorities capable of issuing credentials bound to Users' qualities and attributes
3. Verifiers – entities checking Users' qualities by verifying genuineness of their credentials and obtaining a subset of Users' attributes

The Issuers and Verifiers sets may be overlapping.

There are two basic functions necessary for the system to accomplish its purpose:

1. issue(Issuer, User): a procedure in which an Issuer verifies Users identity, qualities and attributes by some means (not necessarily specified by the function) and, if the verification passes, outputs a credential bound to aforementioned values (which the User stores).
2. verify(User, Verifier): a procedure in which a User proves to a Verifier that a credential which the User possesses has truly been issued by a specified Issuer and the User is its genuine owner (i.e. the credential has not been lent or stolen). In addition, the Verifier checks if the credential has not expired. The output of this procedure is a boolean value denoting if the verification was successful or not (if the credential is valid or not).

The credentials in a system must be *unforgeable*, i.e. no entity besides an Issuer should be able to issue a credential which can be successfully verified as this Issuer's credential, even if any[1] number of valid credentials is available in the forging process. The credential system is *anonymous* if verification procedure does not reveal User's identity. Additionally, the procedures should have *unlinkability* property: two presentations of the same credential should not be easy to connect and the same holds for a presentation and an issuance.

2.2 Zero-Knowledge Proof of Knowledge

A zero-knowledge proof of knowledge is a protocol between two PPT entities called Prover and Verifier. The Prover proves to the Verifier that she knows a set of values without revealing these values. ZKP possesses three properties:

1. Completeness (Correctness) – If a Prover is honest and knows the values being proved, an honest Verifier will always get convinced that the Prover knows these values.
2. Soundness – If a Prover does not know the values being proved, a Verifier will get convinced that the Prover knows these values with at most negligible probability. Moreover, there exists a PPT algorithm called extractor which, having unlimited access to Prover's output and input (including randomness tape), extracts values fulfilling Prover's commitment with probability not smaller than probability of convincing the Verifier by the Prover (extractor existence assures that the Prover possesses the knowledge as it can be extracted based on Prover's behavior).
3. Zero-knowledge – There exists a PPT simulator which, given a statement to be proved, generates a fake proof transcript that is indistinguishable from transcripts which would be generated by honest Verifier and Prover following the protocol. The only entity gaining knowledge is a Verifier and he learns nothing more than a fact that a Prover possesses a value or a set of values she commits to. This knowledge is not transferable; because of simulator existence, genuineness of the information is proved only for the participants which can actively influence the protocol run. Consequently, a transcript itself is not a proof that the Prover possesses the values she committed to.

In this paper, we use Camenisch-Stadler notation [5] for high-level description of zero-knowledge proofs, i.e. proofs are denoted with the template:

$$ZKP\{(\text{Prover's values}) : \text{statement}\}$$

The Prover proves that she possesses such values that the statement holds.

[1]By *any* we mean any polynomial-bound in terms of security parameter λ.

2.3 Bilinear Maps

Let $(\mathbb{G}, +)$ be a group with generator G and $(\mathbb{G}_T, \cdot)$ be a group with generator $\widehat{g}$, such that $|\langle G \rangle| = |\langle \widehat{g} \rangle| = q$. Let $\mathrm{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an efficiently computable bilinear mapping such that:

1. $(\forall A, B \in \mathbb{G})(\forall x, y \in \mathbb{Z}_q)\ \mathrm{e}([x]A, [y]B) = e(A, B)^{xy}$
2. e is non-degenerate, i.e. $\exists\ P, Q$ s.t. $\mathrm{e}(P, Q) \neq 1$, where 1 is the identity element of $\mathbb{G}_T$.

We call the tuple $(\mathbb{G}, +, G, \mathbb{G}_T, \cdot, \widehat{g}, q, \mathrm{e})$ a *bilinear pairing system.*

2.4 CL-LRSW Signatures

We base our construction on CL-LRSW randomizable signatures presented by Camenisch and Lysyanskaya [4]. The signatures are based on LRSW Assumption [13].

ASSUMPTION 2.1. *LRSW Assumption[13] Let $\mathbb{G}$ be a cyclic group with generator G and prime order q. Let $A = [a]G, B = [b]G \in \mathbb{G}$. Let $O_{A,B}(\cdot)$ be an oracle that on input $m \in \mathbb{Z}_q$ outputs $(R, [b]R, [a + mab]R)$ where R is a random element of $\mathbb{G}$.*

Then, having access to such oracle and knowing all values but a and b, it is infeasible to generate a triple (X, Y, Z) such that $X \in \mathbb{G} \wedge Y = [b]X \wedge Z = [a + m'ab]X$ and m' has not been queried to the oracle.

In Figure 1, we recall the signature scheme for blocks of $l+1$ messages (Scheme C, cf. [4]). In said paper, Camenisch and Lysyanskaya also show how to obtain a signature on perfectly hidden values committed as exponents and how to prove that the signature is valid without revealing these values in interactive protocol. With this scheme, a User can obtain a signature under her private values in such a way that even a computationally unbound Issuer cannot learn them. Knowledge of these values is crucial in the signature verification protocol. Hiding values prevents the Issuer from impersonating a User and successfully performing verification procedure with a Verifier. However, the original interactive verification protocol may consume a lot of computational power because it requires computing multiple bilinear mappings by both sides and this number depends on the number of messages signed.

2.5 Fiat-Shamir Transform

Let P be a three-move Zero-Knowledge Proof (ZKP) protocol involving two parties: a Prover and a Verifier, with three steps: Commitment CMT, Challenge CH and Response RSP. One can view this protocol as a set of sequential algorithms:

- GEN-CMT(K) $\to$ (CMT, K') executed by the Prover, where K and K' represent the Prover's internal knowledge.
- GEN-CH(CMT) $\to$ (CH) executed by the Verifier
- GEN-RSP(CH, K') $\to$ (RSP) executed by the Prover
- VERIFY(STMT, CMT, CH, RSP) $\to$ **bool** executed by the Verifier, where STMT is the statement being proved.

Note that Σ-protocols are a wide range of ZKPs, that fall under these requirements.

SETUP (λ):
1: Produce a cryptographic bilinear pairing system $(\mathbb{G}, +, G, \mathbb{G}_T, \cdot, \hat{g}, q, \mathrm{e})$ with security parameter λ.
2: **return** $(\mathbb{G}, +, G, \mathbb{G}_T, \cdot, \hat{g}, q, \mathrm{e})$

GEN (l):
1: $(x, y, z_1, \ldots z_l) \xleftarrow{\$} \mathbb{Z}_q^{l+2}$
2: $X := [x]G$
3: $Y := [y]G$
4: $Z_i := [z_i]G$ for $1 \leq i \leq l$
5: $sk := (x, y, z_1, \ldots z_l)$
6: $pk := (X, Y, Z_1, \ldots Z_l)$
7: **return** (sk, pk)

SIGN$_{sk}$ $\left((m_0, \ldots m_l) \in \mathbb{Z}_q^{l+1}\right)$:
1: $sk = (x, y, z_1, \ldots z_l)$
2: $A_0 \xleftarrow{\$} \mathbb{G}$
3: $B_0 := [y]A_0$
4: **for** $i \in \{1, \ldots l\}$ **do**
5: $\quad A_i := [z_i]A_0$
6: $\quad B_i := [z_i]B_0$
7: $C := [x]A_0 + \sum_{i=0}^{l} [xym_i]A_i$
8: $\sigma := (A_0, \ldots A_l, B_0, \ldots B_l, C)$
9: **return** σ

SIGN$_{sk}$ $(M \in \mathbb{G})$:
1: $sk = (x, y, z_1, \ldots z_l)$
2: $\alpha \xleftarrow{\$} Zq$
3: $A_0 := [\alpha]G$
4: $B_0 := [y]A_0$
5: **for** $i \in \{1, \ldots l\}$ **do**
6: $\quad A_i := [z_i]A_0$
7: $\quad B_i := [z_i]B_0$
8: $C := [x]A_0 + [\alpha xy]M$
9: $\sigma := (A_0, \ldots A_l, B_0, \ldots B_l, C)$
10: **return** σ

VER$_{pk}$ $\left(\sigma, (m_0, \ldots m_l) \in \mathbb{Z}_q^{l+1}\right)$:
1: $pk = (X, Y, Z_1, \ldots Z_l)$
2: $\sigma = (A_0, A_1, \ldots A_l, B_0, B_1, \ldots B_l, C)$
3: **if** $\mathrm{e}(Y, A_0) \neq \mathrm{e}(G, B_0)$ **then return false**
4: **for** $i \in \{1, \ldots l\}$ **do**
5: $\quad$ **if** $\mathrm{e}(A_0, Z_i) \neq \mathrm{e}(G, A_i)$ **then return false**
6: $\quad$ **if** $\mathrm{e}(Y, A_i) \neq \mathrm{e}(G, B_i)$ **then return false**
7: $\widehat{c} := \mathrm{e}(X, A_0) \cdot \prod_{i=0}^{l} \mathrm{e}(X, [m_i]B_i)$
8: **return** $\mathrm{e}(G, C) \stackrel{?}{=} \widehat{c}$

Figure 1: CL-LRSW Signature Scheme C (the first signing algorithm accepts plain massages, while the second one can be used for signing commitment over messages)

Having such a proof protocol, one can transform it into a Non-Interactive Zero-Knowledge (NIZK) scheme, by substituting the GEN-CH(CMT) $\to$ (CH) algorithm with a Random Oracle H. This way, the Verifier does not need to be an active party in the protocol producing the transcript (CMT, CH, RSP), without sacrificing the soundness property

of the scheme. What is more, the tuple (STMT, CMT, RSP) is enough to run the verification algorithm, as CH can be efficiently computed "on fly" as $H(\text{CMT})$.

One can transform the protocol even further, changing it into a signature scheme, by adding message argument MSG as an additional input to the Random Oracle e.g. as $H(\text{MSG}, \text{CMT})$. Thorough analysis of the Fiat-Shamir transform can be found in [1, 9].

3. OUR CONSTRUCTION

In this section, we present the generic core of our improved construction, we later use to create Anonymous Credential system and Domain Certificate system.

3.1 Improved Verification Procedure

Our main contribution to the CL-LRSW construction [4] is the following, simplified verification procedure. We have managed to completely eradicate computation of bilinear pairings on the side of the Prover, while maintaining the same level of security as the original CL-LRSW construction.

As in the original construction, both the Prover and the Verifier have access to an external public key $pk = (X, Y, Z_1, \dots Z_l)$. Prover, holding an LRSW signature $\sigma = (A_0, A_1, \dots A_l, B_0, B_1, \dots B_l, C)$ proves its possession, along with the knowledge of a set of messages $m = (m_0, \dots m_l)$ for which the signature verifies. Cf. Figure 1.

We present our modified construction in Figure 2.

3.1.1 Zero-Knowledge Properties of the Verification Procedure

The verification procedure is in fact a zero-knowledge proof of knowledge of the signature and the message signed. In high-level terms, we can define the protocol for a public key pk as defined in Figure 1 as:

$$ZKP\{(\varsigma, (\mu_0, \dots \mu_l)) : \text{VER}_{pk}(\varsigma, (\mu_0, \dots \mu_l)) = \mathbf{true}\} \quad (1)$$

or in details, for $pk = (X, Y, Z_1, \dots Z_l)$:

$$ZKP\Big\{\Big((\alpha_0, \dots \alpha_l, \beta_0, \dots \beta_l, \gamma), (\mu_0, \dots \mu_l)\Big) : \begin{aligned} &\mathrm{e}(G,\gamma) = \mathrm{e}(X,\alpha_0) \cdot \prod_{i=0}^{l} \mathrm{e}(X,[\mu_i]\beta_i) \\ &\wedge \forall_{1\le i\le l}\ \mathrm{e}(Z_i,\alpha_0) = \mathrm{e}(G,\alpha_i) \\ &\wedge \forall_{0\le i\le l}\ \mathrm{e}(Y,\alpha_i) = \mathrm{e}(G,\beta_i)\Big\} \end{aligned} \quad (2)$$

THEOREM 3.1. *The Improved Verification Protocol (Figure 2) is a zero-knowledge protocol satisfying the proof of knowledge from Equation* (1).

PROOF. In order to prove the aforementioned theorem, we prove that our protocol retains the three ZKP properties:

Correctness.

The protocol from Figure 2 realizes the ZKP from Equation (1), that is, having a signature $\varsigma = (\alpha_0, \dots \alpha_l, \beta_0, \dots \beta_l, \gamma)$ and a message $\mu = (\mu_0, \dots \mu_l)$, satisfying

$$\begin{aligned} &\mathrm{e}(G,\gamma) = \mathrm{e}(X,\alpha_0) \cdot \prod_{i=0}^{l} \mathrm{e}(X,[\mu_i]\beta_i) \\ &\wedge \forall i \in \{1,\dots l\} \quad \mathrm{e}(Z_i,\alpha_0) = \mathrm{e}(G,\alpha_i) \\ &\wedge \forall i \in \{0,\dots l\} \quad \mathrm{e}(Y,\alpha_i) = \mathrm{e}(G,\beta_i), \end{aligned}$$

an honest Prover will always convince an honest Verifier. Let $\sigma = \varsigma$ and $m = \mu$. Therefore, $\sigma' = (\widetilde{A_0}, \dots \widetilde{A_l}, \widetilde{B_0}, \dots \widetilde{B_l}, \widetilde{C}) = ([r']\alpha_0, \dots [r']\alpha_l, [r']\beta_0, \dots [r']\beta_l, [rr']\gamma)$. Analogously, $T = [k_r r']\alpha_0 + \sum_{i\in\{0,\dots l\}} [k_i r']\beta_i$. The first verification statement expands to $\mathrm{e}(G,[c]\widetilde{C}) = \mathrm{e}(G,[crr']\gamma) = \mathrm{e}(G,\gamma)^{crr'}$ on the left-hand side and:

$$\begin{aligned} &\mathrm{e}\Big(X, T - [s_r]\widetilde{A_0} - \sum_{i=0}^{l}[s_i]\widetilde{B_i}\Big) = \\ &\mathrm{e}\Big(X, [k_r r']\alpha_0 + \sum_{i=0}^{l}[k_i r']\beta_i \\ &\quad - [k_r r' - crr']\alpha_0 - \sum_{i=0}^{l}[k_i r' - cr\mu_i r']\beta_i\Big) = \\ &\mathrm{e}\Big(X, [crr']\alpha_0 + \sum_{i=0}^{l}[cr\mu_i r']\beta_i\Big) = \\ &\mathrm{e}\Big(X, [crr']\Big(\alpha_0 + \sum_{i=0}^{l}[\mu_i]\beta_i\Big)\Big) = \\ &\mathrm{e}\Big(X, \alpha_0 + \sum_{i=0}^{l}[\mu_i]\beta_i\Big)^{crr'} = \\ &\Big(\mathrm{e}\Big(X,\alpha_0\Big) \cdot \prod_{i=0}^{l}\mathrm{e}\Big(X,[\mu_i]\beta_i\Big)\Big)^{crr'} \end{aligned}$$

on the right-hand side. Thus, from the equality of the exponents and the fact, that $\mathrm{e}(G,\gamma) = \mathrm{e}(X,\alpha_0)\cdot\prod_{i=0}^{l}\mathrm{e}(X,[\mu_i]\beta_i)$, both sides of the equation are equal. The remaining equations can be verified in a very straightforward way, as ($\forall i \in \{1,\dots l\}$):

$$\begin{aligned} \mathrm{e}(Z_i,\widetilde{A_0}) &= \mathrm{e}(G,\widetilde{A_i}) \\ \mathrm{e}(Z_i,[r']\alpha_0) &= \mathrm{e}(G,[r']\alpha_i) \\ \mathrm{e}(Z_i,\alpha_0)^{r'} &= \mathrm{e}(G,\alpha_i)^{r'} \\ \mathrm{e}(Z_i,\alpha_0) &= \mathrm{e}(G,\alpha_i) \end{aligned}$$

and analogously, $\forall i \in \{0,\dots l\}$:

$$\begin{aligned} \mathrm{e}(Y,\widetilde{A_i}) &= \mathrm{e}(G,\widetilde{B_i}) \\ \mathrm{e}(Y,\alpha_i)^{r'} &= \mathrm{e}(G,\beta_i)^{r'} \\ \mathrm{e}(Y,\alpha_i) &= \mathrm{e}(G,\beta_i). \end{aligned}$$

Soundness.

We base our soundness proof on the existence of a knowledge extractor, that given a black-box algorithm $\mathcal{A}$ successfully performing the verification protocol with a non-negligible probability, extracts all the witnesses. The extractor algorithm runs $\mathcal{A}$ twice with the same random tape, but with different challenges $c \neq c'$, producing two distinct transcripts:

$$\begin{aligned} &(\sigma',\ T,\ c,\ (s_r,\ s_0, \dots\ s_l)) \\ &(\sigma',\ T,\ c',\ (s'_r,\ s'_0, \dots\ s'_l)) \end{aligned}$$

that verify with non-negligible probability. In such a case, we can write:

$$\mathrm{e}(G,[c]\widetilde{C}) = \mathrm{e}\left(X, T - [s_r]\widetilde{A_0} - \sum_{i=0}^{l}[s_i]\widetilde{B_i}\right)$$

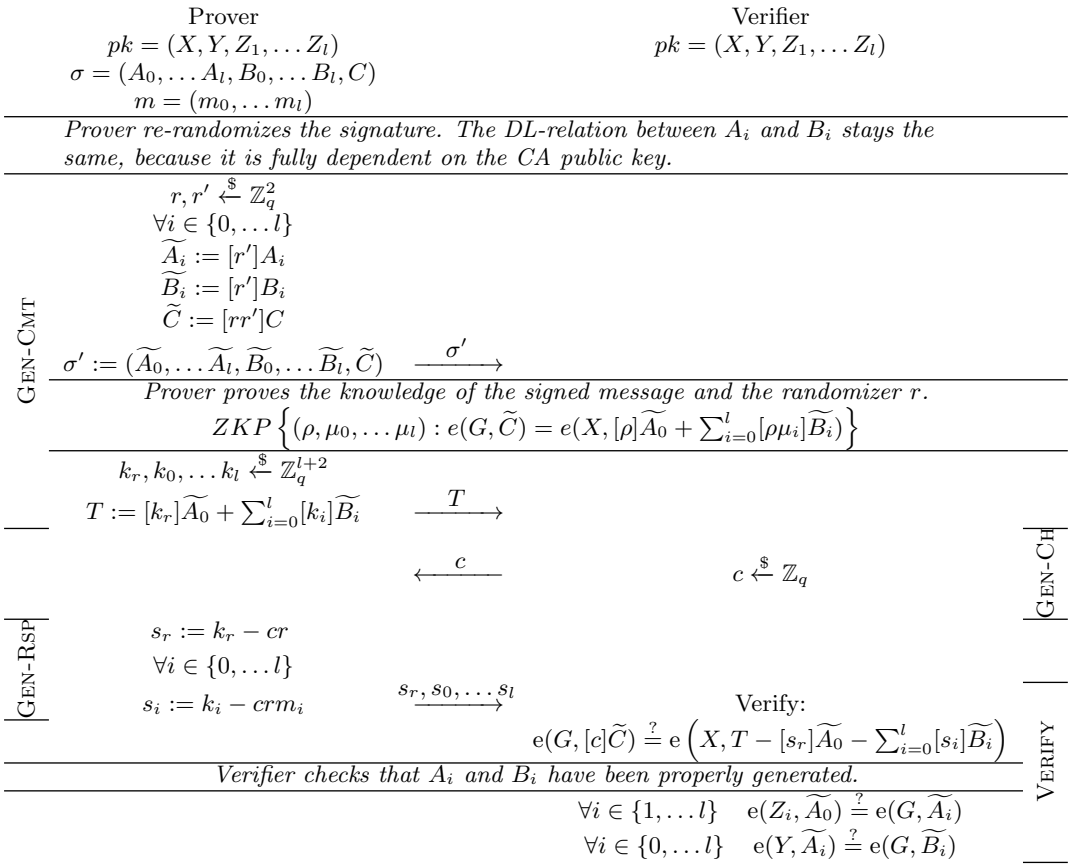

Figure 2: Improved Verification Protocol

$$e(G, [c']\widetilde{C}) = e\left(X, T - [s'_r]\widetilde{A_0} - \sum_{i=0}^{l}[s'_i]\widetilde{B_i}\right)$$

where $\sigma' = (\widetilde{A_0}, \dots \widetilde{A_l}, \widetilde{B_0}, \dots \widetilde{B_l}, \widetilde{C})$. From the equation we derive:

$$e(G, [c]\widetilde{C}) = e(X, T)\, e\left(X, -[s_r]\widetilde{A_0} - \sum_{i=0}^{l}[s_i]\widetilde{B_i}\right)$$

$$e(G, [c']\widetilde{C}) = e(X, T)\, e\left(X, -[s'_r]\widetilde{A_0} - \sum_{i=0}^{l}[s'_i]\widetilde{B_i}\right)$$

which in turn equals to

$$e(X, T) = e(G, [c]\widetilde{C})\ e\left(X, [s_r]\widetilde{A_0} + \sum_{i=0}^{l}[s_i]\widetilde{B_i}\right)$$

$$e(X, T) = e(G, [c']\widetilde{C})\ e\left(X, [s'_r]\widetilde{A_0} + \sum_{i=0}^{l}[s'_i]\widetilde{B_i}\right).$$

When we merge the equations, we get

$$e(G, [c]\widetilde{C})\ e\left(X, [s_r]\widetilde{A_0} + \sum_{i=0}^{l}[s_i]\widetilde{B_i}\right)$$
$$= e(G, [c']\widetilde{C})\ e\left(X, [s'_r]\widetilde{A_0} + \sum_{i=0}^{l}[s'_i]\widetilde{B_i}\right)$$

and after reordering

$$e(G, [c]\widetilde{C})/e(G, [c']\widetilde{C}) =$$
$$e\left(X, [s'_r]\widetilde{A_0} + \sum_{i=0}^{l}[s'_i]\widetilde{B_i}\right)/e\left(X, [s_r]\widetilde{A_0} + \sum_{i=0}^{l}[s_i]\widetilde{B_i}\right)$$

which can be simplified to

$$e(G, [c - c']\widetilde{C}) = e\left(X, [s'_r - s_r]\widetilde{A_0} + \sum_{i=0}^{l}[s'_i - s_i]\widetilde{B_i}\right)$$

$$e(G, \widetilde{C})^{c-c'} = e\left(X, [s'_r - s_r]\widetilde{A_0} + \sum_{i=0}^{l}[s'_i - s_i]\widetilde{B_i}\right).$$

When we split the sum inside the bilinear operator and change point multiplication into exponentiation, we acquire the verification statement:

$$e(G, \widetilde{C}) = e(X, \widetilde{A_0})^{(s'_r - s_r)/(c-c')} \cdot \prod_{i=0}^{l} e(X, \widetilde{B_i})^{(s'_i - s_i)/(c-c')}.$$

It does not, however, correspond directly to the statement present in the ZKP form. Let us denote $\rho = (s'_r - s_r)/(c - c')$, $\widetilde{\mu_i} = (s'_i - s_i)/(c - c')$ and $\mu_i = \widetilde{\mu_i}/\rho = (s'_i - s_i)/(s'_r - s_r)$. Then, by exponentiating both sides of the equation by $1/\rho$, we obtain:

$$e(G, [1/\rho]\widetilde{C}) = e(X, \widetilde{A_0}) \cdot \prod_{i=0}^{l} e(X, \widetilde{B_i})^{\mu_i}.$$

which is equivalent to the statement from Equation (2) with $\gamma = [1/\rho]\widetilde{C}$, $\alpha_i = \widetilde{A_i}$, $\beta_i = \widetilde{B_i}$ and μ_i extracted as above. Note that the rest of the statement (namely the $\forall \ldots$ cases) follows directly from the last part of the verification protocol.

Zero-Knowledge.

We base our zero-knowledge property on a simulator, that given one CL-LRSW signature under *any* message, generates a fake proof transcript that is indistinguishable from an honestly generated one. The assumption of existence of at least one CL-LRSW signature does not compromise security properties, as the unforgeability proof explicitly assumes the existence of an oracle generating valid LRSW signatures.

The simulator is given in Figure 3. No entity, even unbound in terms of computational power, can distinguish the output of the SIMULATE function, from a genuine protocol transcript, since:

1. in simulation: $s_r, s_0, \ldots s_l$ are taken uniformly at random from a cyclic group; in real transcript: $s_x = k_x - c(\text{any value})$ with k_x taken uniformly at random, independent from c and (any value); from the properties of cyclic groups, regardless of distribution of c and (any value), both values have the same distributions as uniform values from $\mathbb{Z}_q$;

2. A_i and $\widetilde{A_i}$ (a randomized value in real protocol, cf. Figure 2) can be viewed as $[r'][\alpha'][z_i]G$ and $[\widetilde{r'}][\alpha][z_i]G$ with $z_0 = 1$ for some values α, α'. It is easy to see, that each of the values can be viewed as $[z_i]G$ multiplied by the same, uniformly distributed value from $\mathbb{Z}_q$;

3. B_i and $\widetilde{B_i}$ can be viewed as $[y]A_i$ and $[y]\widetilde{A_i}$ respectively, thus, their distribution is the same and their dependency to A_i ($\widetilde{A_i}$) is preserved;

4. C can be viewed as a completely random element of G; Since $\widetilde{C_i}$ is a value randomized with uniformly chosen $\mathbb{Z}_q$ element, they have the same, uniform distribution.

5. in simulation, $T = [c/x]C + [s_r]A_0 + \sum_{i=0}^{l}[s_i]B_i$; in real transcript, $\mathrm{e}(X, [c/x]\widetilde{C}) = \mathrm{e}(G, [c]\widetilde{C}) = \mathrm{e}\big(X, T - [s_r]\widetilde{A_0} - \sum_{i=0}^{l}[s_i]\widetilde{B_i}\big)$ holds; thus, from bilinear properties of e and properties of cyclic groups, $[c/x]\widetilde{C} = T - [s_r]\widetilde{A_0} - \sum_{i=0}^{l}[s_i]\widetilde{B_i}$, therefore in both cases, the values are completely fixed (and equal) in terms of the remaining values.

□

3.2 Change-Base Step

In this section, we present another primitive that we call Change-Base step. It is an additional procedure, that can be mixed with a Σ-protocol ZKP of a special form, to greatly reduce the number of steps necessary to prove equality of any number of entangled values.

Take a simple, classic case of

$$ZKP\left\{(\alpha, \beta, \gamma) : A = x^{\alpha}y^{\beta}z^{\gamma} \wedge B = g^{\alpha}h^{\beta} \wedge C = d^{-\gamma}\right\}.$$

It is relatively easy to make a three-move Σ-protocol satisfying the ZKP, by taking three random values $k_\alpha, k_\beta, k_\gamma$, one for each of the exponents, committing to them in all three base-cases: $t_A = x^{k_\alpha}y^{k_\beta}z^{k_\gamma}, t_B = g^{k_\alpha}h^{k_\beta}, t_C = d^{-k_\alpha}$ and, after receiving a single challenge c, computing the corresponding responses $s_\alpha = k_\alpha - c\alpha, s_\beta = k_\beta - c\beta, s_\gamma = k_\gamma - c\gamma$. The verification step is rather straightforward and involves checking one equation for each of the initial statements: $A^c = t_A x^{s_\alpha}y^{s_\beta}z^{s_\gamma}, B^c = t_B g^{s_\alpha}h^{s_\beta}, C^c = t_C d^{-s_\gamma}$ [15].

In our scenario, the situation is much more complicated, as we have an equation of a form (if we take only the second parameter of the bilinear equation and move the $[x]$ factor from $\widetilde{C}$ to G in order to align first parameters) $[1/x]\widetilde{C} = [r]\widetilde{A_0} + \sum_{i=0}^{l}[rm_i]\widetilde{B_i}$, while sometimes, we would like to reveal elements of a form, for instance $w = g^{m_0}h^{m_1}$. A straightforward proof requires to first show the equality of exponents in case of an intermediate value $v = g^{rm_0}h^{rm_1}$ and then again with the same v being equal to w^r. This still can be done in a single proof, but requires additional steps and intermediate values.

Instead, we propose a generic algorithm that allows to efficiently prove equation of exponents of values with different bases, when, in the original equation, the exponents are entangled with another exponent, existing in that (or actually any other) equation.

SIMULATE (c, pk, σ'):
1: $(X, Y, Z_1, \ldots Z_l) = pk$
2: $(A'_0, A'_1, \ldots A'_l, B'_0, B'_1, \ldots B'_l, C') = \sigma'$ ▷ We discard C', as it corresponds to an unknown message vector m.
3: $r, r', s_r, s_0, \ldots s_l \xleftarrow{\$} Z_q^{l+4}$
4: $C = [r]X$
5: $\overline{C} = [r]G$
6: **for** $i \in \{0, \ldots l\}$ **do**
7: $\quad A_i = [r']A'_i$
8: $\quad B_i = [r']B'_i$
9: $T = [c]\overline{C} + [s_r]A_0 + \sum_{i=0}^{l}[s_i]B_i$
10: $\sigma = (A_0, \ldots A_l, B_0, \ldots B_l, C)$
11: **return** $(\sigma, T, c, (s_r, s_0, \ldots s_l))$

Figure 3: Improved verification protocol simulator (for a challenge c, public key pk and any valid signature σ')

THEOREM 3.2. *Assume we have* $ZKP\{(r, \{\alpha_i\}_{i\in L}, \{\beta_i\}) : \bigwedge E\}$ *with a set of indexed exponents L and a set of equations E. Assume we have a correct, simulable and sound Σ-protocol* GEN-CMT, GEN-CH, GEN-RSP, VERIFY *performing such a proof, with values $s_r = k_r - cr$ and $s_i = k_i - cr\alpha_i$ for all $i \in L$ computed in the* GEN-RSP *step (for $k_r, k_i \ldots$ computed as part of K' in* GEN-CMT*) and knowledge extractors $\alpha_i = (s_i - s'_i)/(s_r - s'_r)$.*

Then we can easily create a Σ-protocol proving

$$ZKP\left\{\left(r, \{\alpha_i\}_{i\in L}, \{\beta_i\}\right) : \bigwedge E \wedge d = \prod_{i\in L} g_i^{\alpha_i}\right\}$$

for any constant set of bases $\{d\} \cup \bigcup_{i\in L}\{g_i\}$ by adding $t = d^{k_r}\prod_{i\in L} g_i^{-k_i}$ to GEN-CMT *step and $t \stackrel{?}{=} d^{s_r}\prod_{i\in L} g_i^{-s_i}$ to* VERIFY.

The produced proof is also correct, sound and simulable.

PROOF. In order to prove the aforementioned theorem, we prove that our protocol retains the three ZKP properties:

Correctness.

Straightforward computation shows, that the protocol is correct:

$$
\begin{aligned}
t &= d^{s_r} \prod_{i \in L} g_i^{-s_i} \\
d^{k_r} \prod_{i \in L} g_i^{-k_i} &= d^{s_r} \prod_{i \in L} g_i^{-s_i} \\
d^{k_r} \prod_{i \in L} g_i^{-k_i} &= d^{k_r - cr} \prod_{i \in L} g_i^{-(k_i - cr\alpha_i)} \\
\left(\prod_{i \in L} g_i^{\alpha_i} \right)^{k_r} \prod_{i \in L} g_i^{-k_i} &= \left(\prod_{i \in L} g_i^{\alpha_i} \right)^{k_r - cr} \prod_{i \in L} g_i^{-(k_i - cr\alpha_i)} \\
\prod_{i \in L} g_i^{\alpha_i k_r} \prod_{i \in L} g_i^{-k_i} &= \prod_{i \in L} g_i^{\alpha_i (k_r - cr)} \prod_{i \in L} g_i^{cr\alpha_i - k_i} \\
\prod_{i \in L} g_i^{\alpha_i k_r - k_i} &= \prod_{i \in L} g_i^{\alpha_i k_r - \alpha_i cr} \prod_{i \in L} g_i^{cr\alpha_i - k_i} \\
\prod_{i \in L} g_i^{\alpha_i k_r - k_i} &= \prod_{i \in L} g_i^{\alpha_i k_r - \alpha_i cr + cr\alpha_i - k_i} \\
\prod_{i \in L} g_i^{\alpha_i k_r - k_i} &= \prod_{i \in L} g_i^{\alpha_i k_r - k_i}
\end{aligned}
$$

Soundness.

It suffices to show, that the same knowledge extractors apply to the given equation. That is, for two different challenge values c and c' and two different sets of response values $\{s_i\}_{i \in L}, s_r$, $\{s'_i\}_{i \in L}, s'_r$ corresponding to the same Prover-commitment t:

$$d^{s_r} \prod_{i \in L} g_i^{-s_i} = t = d^{s'_r} \prod_{i \in L} g_i^{-s'_i}.$$

From this, we derive

$$
\begin{aligned}
d^{s_r} / d^{s'_r} &= \prod_{i \in L} g_i^{-s'_i} / \prod_{i \in L} g_i^{-s_i} \\
d^{s_r - s'_r} &= \prod_{i \in L} g_i^{s_i - s'_i} \\
d &= \prod_{i \in L} g_i^{\frac{s_i - s'_i}{s_r - s'_r}}
\end{aligned}
$$

Thus, we can extract the individual exponents of form $\alpha_i = (s_i - s'_i)/(s_r - s'_r)$ s.t. $d = \prod_{i \in L} g_i^{\alpha_i}$.

Simulability.

Simulating t is straightforward and can be easily derived from the verification statement. The distribution of t in case of the simulation is equal to the distribution in case of honest execution. □

It should also be mentioned, that the verification step can be changed into an *extraction* step, changing verification of d into its extraction, that is *extracting* d from the commitment value t and response values s_r, s_i:

$$
\begin{aligned}
d^{s_r} \prod_{i \in L} g_i^{-s_i} &= t \\
d^{s_r} &= t \prod_{i \in L} g_i^{s_i} \\
d &= \left(t \prod_{i \in L} g_i^{s_i} \right)^{1/s_r}.
\end{aligned}
$$

This trick is very useful to reduce communication in some scenarios.

4. ANONYMOUS CREDENTIALS SYSTEM DESCRIPTION

In this section we present modified CL-LRSW anonymous credential system in details. Note that most of the steps are the same in the original system since our modification affects credential verification protocol. We present the whole system description for readability.

Let $(\mathbb{G}, +, G, \mathbb{G}_T, \cdot, \hat{g}, q, \mathrm{e})$ be an output from $\textsc{Setup}(\lambda)$ function defined in Figure 1. Note that $\mathrm{e}(\cdot, \cdot)$ is a bilinear mapping operation in symmetric pairing, that maps elements from $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and $\hat{g}$ is the image of mapping $e(G, G)$.

The construction can be easily adjusted to asymmetric pairings with no loss of security properties. In this paper, we do not provide the exact protocols for asymmetric pairings due to the space constraints.

4.1 Entities Definition

Let $l - 1$ be a number of attributes for a fixed Issuer. Let $sk_I = (x, y, z_1, \ldots z_l)$ be the Issuer's private key and $pk_I = (X, Y, Z_1, \ldots Z_l)$ a corresponding public key, cf. $\textsc{Gen}(l)$ function defined in Figure 1.

Let a User posses an ordered set of $l-1$ attributes $(m_i : i \in \{2, \ldots l\})$. Let a User's private key be an additional attribute denoted as m_1 such that $m_1 \in \mathbb{Z}_q$. Let $pk_U = [m_1]G$ be User's public key.

Let all Verifiers know all Issuers' public keys. This can be achieved with, for example, public key certificates infrastructure and it is out of scope of this paper.

4.2 Functions Definition

Credential Issuance.

A User approaches an Issuer, divides her attributes into two sets: hidden and public. The User generates a random value $m_0 \in \mathbb{Z}_q$ which could be understood as an attribute with index 0 which is always hidden. The User's private key is also one of the hidden attributes. Let P denote the set of the indices of public attributes and H denote the remaining, hidden attributes indices. Next, the User sends to the Issuer $M_H = \sum_{i \in H} [m_i] Z_i$ and her public attributes list.

The generic version of issuance protocol is presented in Figure 4. The Issuer has to verify if all the public attributes are valid for the User. Possession of the hidden attributes is proved in ZKP. It may happen that some of the hidden attributes possess additional properties, for instance, the generator raised to an exponent of private key value yields a public key. Existence of such properties of hidden values can be proven with the Change-Base construction (cf. Section 3.2). As a result, additional commitment should be created for the private key and sent to the Issuer for verification.

If the verification passes, an Issuer makes a CL-LRSW signature $\textsc{Sign}_{sk_I}\left(M_H + \sum_{i \in P} [m_i] Z_i\right)$ and sends it to the User. The User stores the signature together with all the attributes.

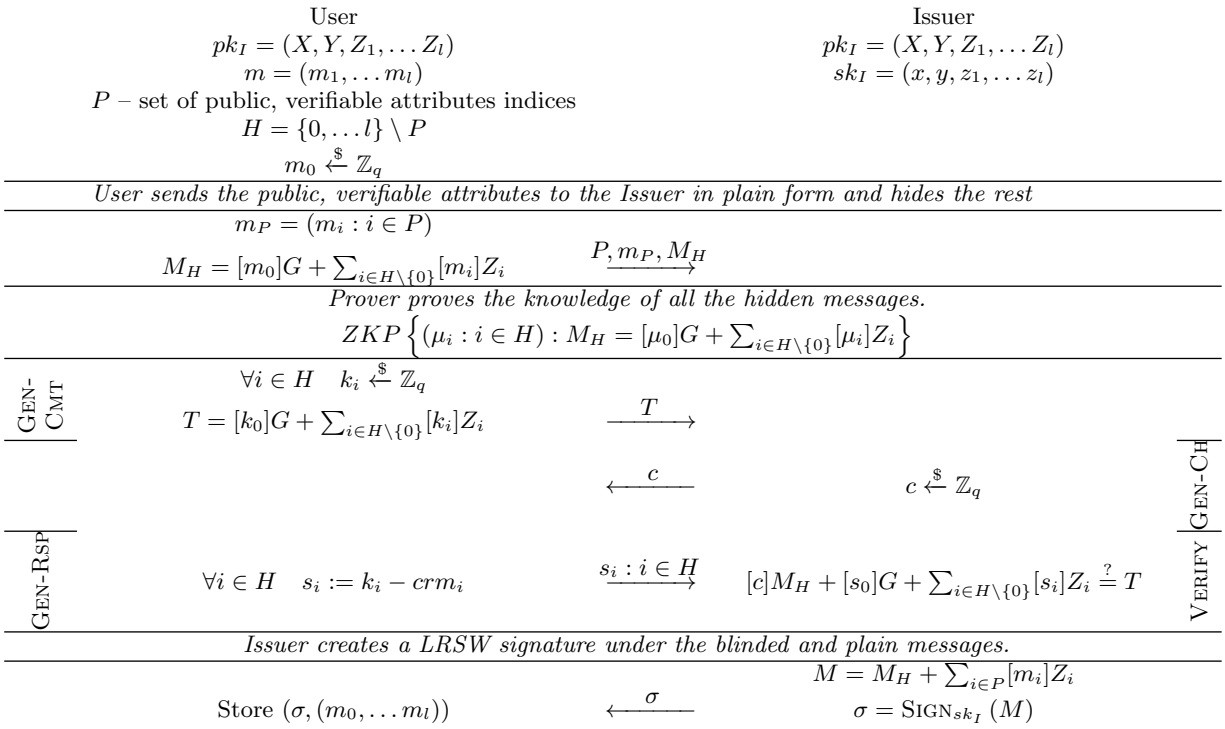

Figure 4: Credential Issuance Protocol

Credential Verification.

A User approaches a Verifier and lets him know that she wants to present a credential issued with the Issuer's public key pk_I. The Verifier checks whether this key is valid (for example, he checks if it has not expired or if such an Issuer is trusted). Then, both parties perform verification protocol described in Figure 2. If the User wants to show additional attributes properties, she modifies ZKP protocol with the Change-Base construction. The output of the verification protocol is **true** if all the Verifier's checks pass.

4.3 Security Properties

The credentials are of the form of CL-LRSW signature C. Since the forgery of such a signature violates assumption of LRSW and discrete logarithm computation hardness, as has been shown in the original paper, the probability of a forgery is negligible for both our and the original system.

As it has been proven in Section 3.1 and Section 3.2, the verification protocol is a zero-knowledge proof of knowledge.

Our system diverges from the CL-LRSW system with verification protocol. Thus, it suffices to prove that security properties for this modified procedure are unaffected by our changes. Since we have shown that our construction is a zero-knowledge proof of possession of unforgeable CL-LRSW signature C, our system is as secure as the original one. All security proofs which are not present in this paper will be the same as in the case of the original scheme.

5. DOMAIN-SPECIFIC PSEUDONYMS FOR SIGNATURES AND IDENTIFICATION

In this section we show how to construct certificate system for domain-(pseudonym-)based schemes, such as pseudonymous signatures and domain identification.

The certificates, as presented in this section, may contain any number of attributes, although only one of them is explicitly used – User's inter-domain secret key, used to create domain-specific public keys, called *pseudonyms*. The same technique can be applied for any number of attributes, depending on the specific application. The issue is explained in more details in Section 5.2

Our construction does not explicitly indicate the signature or identification scheme. Instead, we leave the details to the specific implementations. Intuitively, almost any secure signature scheme which takes $(\langle g \rangle, g, q)$ as system parameters and $sk, pk = g^{sk}$ as User's key pair for g – a generator of a cyclic group of order q and $sk \in \mathbb{Z}_q$, can be transformed into a domain signature scheme using our certificate system. Analogously, having such an identification scheme, a pseudonymous identification scheme can be created. For each such an attempt, an individual security proof should be performed, as the signature and identification schemes themselves use various proof techniques and no general rule can be constructed. However, the security of our scheme holds for Schnorr Signatures and Schnorr Identification Protocol.

We strongly base our construction on the construction of anonymous credentials from Section 4, applying Fiat-Shamir transform on the ZKP from Equation (1) in order to provide static signatures under the certificates.

System Setup.

Let $(\mathbb{G}, +, G, \mathbb{G}_T, \cdot, \hat{g}, q, \mathrm{e})$ be a set of pairing-friendly domain parameters with $\mathrm{e}(\cdot, \cdot)$ being a bilinear mapping operation, mapping elements from $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and $\hat{g}$ being image of mapping $\mathrm{e}(G, G)$. Let $(\mathbb{G}_D, \cdot, q, \widetilde{g})$ be another cyclic group of prime order q, with generator $\widetilde{g}$.

ReCert (m, σ, ξ):
1: $(A_0, \dots A_l, B_0, \dots B_l, C) = \sigma$
2: $(m_0, \dots m_l) = m$
3: $r, r' \xleftarrow{\$} \mathbb{Z}_q^2$
4: **for** $i \in \{0, \dots l\}$ **do**
5: $\quad \widetilde{A_i} := [r']A_i$
6: $\quad \widetilde{B_i} := [r']B_i$
7: $\widetilde{C} := [rr']C$
8: $k_r, k_0, \dots k_l \xleftarrow{\$} \mathbb{Z}_q^{l+2}$
9: $T := [k_r]\widetilde{A_0} + \sum_{i=0}^{l}[k_i]\widetilde{B_i}$
10: $t_{pk} := \xi^{m_1 k_r - k_1}$
11: $e := H(T||t_{pk}||\widetilde{A_0}|| \dots \widetilde{A_l}||\widetilde{B_0}|| \dots \widetilde{B_l}||\widetilde{C})$
12: $s_r := k_r - er$
13: **for** $i \in \{0, \dots l\}$ **do**
14: $\quad s_i := k_i - erm_i$
15: $\sigma' := (\widetilde{A_0}, \dots \widetilde{A_l}, \widetilde{B_0}, \dots \widetilde{B_l}, \widetilde{C})$
16: $S := (s_r, s_0, \dots s_l)$
17: **return** (σ', S, T, t_{pk})

Figure 5: Creating domain-specific certificate

CertVerify $((\sigma, S, T, t_{pk}), \xi, pk_{CA})$:
1: $(A_0, \dots A_l, B_0, \dots B_l, C) = \sigma$
2: $(s_r, s_0, \dots s_l) = S$
3: $e := H(T||t_pk||A_0|| \dots A_l||B_0|| \dots B_l||C)$
4: **if** $\mathrm{e}(G, [e]\widetilde{C}) \neq \mathrm{e}\left(X, T - [s_r]A_0 - \sum_{i=0}^{l}[s_i]B_i\right)$ **then**
5: $\quad$ **return** (**false**, $\perp$)
6: **for** $i \in \{1, \dots l\}$ **do**
7: $\quad$ **if** $\mathrm{e}(Z_i, \widetilde{A_0}) \neq \mathrm{e}(G, \widetilde{A_i})$ **then return** (**false**, $\perp$)
8: **for** $i \in \{0, \dots l\}$ **do**
9: $\quad$ **if** $\mathrm{e}(Y, \widetilde{A_i}) \neq \mathrm{e}(G, \widetilde{B_i})$ **then return** (**false**, $\perp$)
10: $pk_{U,\xi} = \left(t_{pk}\xi^{s_1}\right)^{1/s_r}$
11: **return** (**true**, $pk_{U,\xi}$)

Figure 6: Verifying domain-specific certificate and extracting public key

5.1 Entities Definition

The *Certificate Authority* (CA) is an entity responsible for issuing certificates in the form of credentials. Let us denote the number of attributes in a certificate (including User's secret key) as $l \geq 1$. CA possesses a key pair which is used for credentials issuance $(sk_{CA}, pk_{CA}) = \mathrm{Gen}\,(l)$. The CA is owner of a *master domain* generated by $\widetilde{g}$.

The *Domain Owners* are entities who own *domains of pseudonyms* (each of them a single one). Let $\xi \in \mathbb{G}_D$ be a generator bound to the domain. The selection and validation of ξ is crucial to the security of the scheme. ξ has to be publicly verifiable, selected pseudorandomly, with no known DL-relation to any previously known element of $\mathbb{G}_D$. This can be achieved by selecting ξ as an output of a hash function, somehow based on domain's common name. Another important issue is to check if Verifier is actually bound to the domain ξ, as deanonymization through presenting rogue domain bases would be trivial. Since there is a number of different solutions to the issue, we leave its specification as a future work. One should only remember that the parameters of a Domain Owner depend on the chosen solution.

The *Users* are entities somewhat similar to credential system Users. Let us denote a User's secret key as sk_U and a corresponding User's public key in *master domain* as $pk_U = \widetilde{g}^{sk_U}$. A User obtains a certificate in a form of a credential from the Certificate Authority. Then, she presents it to the Domain Owner under domain pseudonym.

5.2 Functions Definition

Certificate Issuance.

The procedure is identical to Credential Issuance from Section 4.2. The first attribute, m_1, is the User's inter-domain secret key sk_U and thus its possession shall be proven using the Change-Base construction. The remaining attributes are not implicitly used in the certificates, as presented in this section, but can be revealed using Credential Verification protocol from Section 4.2 or as additional fields in the certificate, via additional Change-Base steps. Let us denote obtained credential as σ. Moreover, let us define procedure IssueCert as a protocol between the Issuer and a User which runs $\mathrm{Sign}_{sk_{CA}}$ on User's attributes.

Certificate Domain-Pseudonymization.

Recall the ZKP from Equation (1). Using the construction from Section 3.2, let us slightly modify the statement, to also prove the equality of exponents for domain pseudonym $pk_{U,\xi} = \xi^{sk_U} = \xi^{m_1}$:

$$ZKP\{(\varsigma, (\mu_0, \dots \mu_l)) : \mathrm{Ver}_{pk}\,(\varsigma, (\mu_0, \dots \mu_l)) = 1 \\ \wedge\, pk_{U,\xi} = \xi^{\mu_1}\}.$$

We apply Fiat-Shamir Transform on the modified ZKP in order to produce a signature of knowledge (SoK), in form of a domain-specific certificate, which allows extracting User's public key in domain ξ. We present the responsible procedure ReCert in Figure 5. The certificate is of a form (σ, S, T, t_{pk}).

Certificate Verification.

While *Certificate Domain-Pseudonymization* produces a domain-specific certificate, this function allows anyone to verify its correctness, extracting the domain-specific public key $pk_{U,\xi}$ bound to it. It is essentially the verification step of the aformetnioned SoK, with a verification part changed into extraction as suggested in Section 3.2. We present the responsible procedure CertVerify in Figure 6. The procedure returns pairs of a form $(\mathbf{true}, pk_{U,\xi})$ upon a successfull verification and $(\mathbf{false}, \perp)$ otherwise.

5.3 Security Properties

5.3.1 Unforgeability

The unforgeability of the certificate issuance follows directly from the unforgeability of anonymous credentials, cf. Section 4.3.

5.3.2 Unlinkability

Definition 5.1. ***Unlinkability Game*** *We define the Unlikability Game between an adversary $\mathcal{A}(\cdot)$ and a challenger $\mathcal{CH}(\cdot)$ as the following experiment:*

1. *$\mathcal{CH}$ on input takes the three groups of order q : $\mathbb{G}, \mathbb{G}_T$ and $\mathbb{G}_D$ with a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, gener-*

CertSimulate (pk_U, g, sk_{CA}):

1: $(x, y, z_1, \dots z_l) = sk_{CA}$
2: $C \xleftarrow{\$} \mathbb{G}$
3: $A_0 \xleftarrow{\$} \mathbb{G}$
4: $B_0 := [y]A_0$
5: **for** $i \in \{1, \dots l\}$ **do**
6: $\quad A_i := [z_i]A_0$
7: $\quad B_i := [z_i]B_0$
8: $\sigma = (A_0, \dots A_l, B_0, \dots B_l, C)$
9: $S = (s_r, s_0, \dots s_l) \xleftarrow{\$} \mathbb{Z}_q^{l+2}$
10: $e \xleftarrow{\$} \mathbb{Z}_q$
11: $T = [1/x]C + [s_r]A_0 + \sum_{i=0}^{l}[s_i]B_i$
12: $t_{pk} := pk_U^{s_r} g^{-s'_1}$
13: $\mathrm{Q} := T||t_{pk}||A_0|| \dots A_l||B_0|| \dots B_l||C$
14: **if** H has been queried for Q **then**
15: $\quad$ **goto** step 10
16: Program $H(\mathrm{Q}) := e$
17: **return** (σ, S, T, t_{pk})

Figure 7: Domain Certificate simulator (programming random oracle H)

ators $G \in \mathbb{G}, \hat{g} \in \mathbb{G}_T, \widetilde{g} \in \mathbb{G}_D$ s.t. $\hat{g} = e(G, G)$, number of attributes l and bit b. We denote it $\mathcal{CH}(b, l)$.

2. *$\mathcal{CH}$ initiates certificate authority $(sk_{CA}, pk_{CA}) = \textsc{Gen}\,(l)$ (cf. Figure 1).*

3. *If $b = 0$, $\mathcal{CH}$ selects two random private keys $sk_U, sk_{U'} \xleftarrow{\$} \mathbb{Z}_q^2$; otherwise $sk_U = sk_{U'} \xleftarrow{\$} \mathbb{Z}_q$.*

4. *$\mathcal{CH}$ generates two domains ζ and ξ as random values from $\mathbb{G}_D$.*

5. *$\mathcal{CH}$ generates two certificates c_ξ, c'_ζ for the corresponding secret keys $sk_U, sk_{U'}$ and domains ξ, ζ. The certificates pass verification, that is:*

$$(\textbf{\textit{true}}, \xi^{sk_U}) = \textsc{CertVerify}(c_\xi, \xi, pk_{CA})$$
$$(\textbf{\textit{true}}, \zeta^{sk'_U}) = \textsc{CertVerify}(c'_\zeta, \zeta, pk_{CA})$$

6. *$\mathcal{A}$ is given the certificates and CA description (including private keys) and outputs bit*

$$b' = \mathcal{A}(c_\xi, c'_\zeta, \xi, \zeta, sk_{CA}, pk_{CA}).$$

7. *We say that the adversary wins the game, if $b = b'$.*

Theorem 5.2. *The certificate scheme* IssueCert, ReCert, CertVerify *is unlinkable under DDH-assumption in $\mathbb{G}_D$, i.e. if DDH problem is hard in $\mathbb{G}_D$ no polynomial time adversary $\mathcal{A}$ wins the Unlinkability Game (Definition 5.1) with more than a negligible advantage:*

$$\Big|\Pr\left[1 = \langle \mathcal{A}|\mathcal{CH}(b, l)\rangle | b = 1\right] - \Pr\left[1 = \langle \mathcal{A}|\mathcal{CH}(b, l)\rangle | b = 0\right]\Big| \leq negl(\lambda)$$

where λ is the groups' security parameter.

Proof. Given an adversary that wins the linking game with a non-negligible advantage ϵ we can construct a simulator in the Random Oracle Model that solves the DDH-problem in $\mathbb{G}_D$. On input DDH tuple (g, g^a, g^b, g^c) we set $\xi = g$, $\zeta = g^a$, $pk_{U,\xi} = g^b$, $pk_{U',\zeta} = g^c$.

We construct a simulator for the challenger in the following way:

1. Initiate certificate authority $(sk_{CA}, pk_{CA}) = \textsc{Gen}\,(l)$.
2. Simulate domain certificates, programming the Random Oracle:

$$c_\xi = \textsc{CertSimulate}(pk_{U,\xi}, \xi, sk_{CA})$$
$$c_\zeta = \textsc{CertSimulate}(pk_{U',\zeta}, \zeta, sk_{CA})$$

3. Run and **return** $\mathcal{A}(c_\xi, c'_\zeta, \zeta, \xi, sk_{CA}, pk_{CA})$

The adversary simply cannot derive any information about the Users' relationship from the certificates (but the public-key parts $pk_{U,\xi}, pk_{U',\zeta}$) because CertSimulate procedure is oblivious to their private keys, or any public keys in $\mathbb{G}$. By solving the unlinkability problem (and winning the unlinkability game), the adversary claims that either:

- both public keys have the same private exponent α i.e. $pk_{U,\xi} = \xi^\alpha$ and $pk_{U',\zeta} = \zeta^\alpha$;
- both public keys have different private exponents α and β i.e. $pk_{U,\xi} = \xi^\alpha$ and $pk_{U',\zeta} = \zeta^\beta$.

In the first case, we obtain $g^b = \xi^\alpha = g^\alpha \Rightarrow b = \alpha$ and $g^c = \zeta^\alpha = (g^a)^b = g^{ab} \Rightarrow c = ab$.

In the second case, we get $g^b = \xi^\alpha = g^\alpha \Rightarrow b = \alpha$ and $g^c = \zeta^\beta = (g^a)^\beta = g^{a\beta} \wedge \alpha \neq \beta \Rightarrow c = a\beta \neq a\alpha = ab \Rightarrow c \neq ab$. □

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a generalized version of verification procedure for Sigature Scheme C of Camenisch and Lysyanskaya [4]. We presented a short and efficient Zero-Knowledge Protocol for this procedure, which requires computing only $2l+3$ bilinear pairings on the side of the Verifier and does not require any bilinear operation or operation in $\mathbb{G}_T$ on the side of the Prover, while proofs based solely on the original construction require $l+2$ pairings on the side of the Prover and $5l+5$ on the side of the Verifier.

We also presented a generic construction allowing efficient extraction of entangled exponents in specific cases. While the construction was designed specifically for the solution, it can be applied to much wider range of Σ-protocols.

Finally, we presented two generic constructions – a slightly modified Anonymous Credential system, tightly based on the original construction by Camenisch and Lysyanskaya – and a Pseudonymous Domain Certificate system – a basic building block for unified Pseudonymous Signatures and Restricted Identification systems. Unfortunately, due to space limitations, we were unable to provide a fully working domain-based system. We leave its specification as a future work.

We still see a room for improvement in efficiency of verification procedures, as many bilinear mappings share partial inputs. We leave this, along with a reference implementation, as a future work.

7. REFERENCES

[1] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *Advances in Cryptology—EUROCRYPT 2002*, pages 418–433. Springer, 2002.

[2] J. Bender, Ö. Dagdelen, M. Fischlin, and D. Kügler. Domain-specific pseudonymous signatures for the german identity card. In *Information Security*, pages 104–119. Springer, 2012.

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

[4] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology–CRYPTO 2004*, pages 56–72. Springer, 2004.

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

[6] D. Chaum. Showing credentials without identification transferring signatures between unconditionally unlinkable pseudonyms. In *Advances in Cryptology—AUSCRYPT'90*, pages 245–264. Springer, 1990.

[7] L. Chen, P. Morrissey, and N. P. Smart. Daa: Fixing the pairing based protocols. *IACR Cryptology ePrint Archive*, 2009:198, 2009.

[8] L. Chen, D. Page, and N. P. Smart. On the design and implementation of an efficient daa scheme. In *International Conference on Smart Card Research and Advanced Applications*, pages 223–237. Springer, 2010.

[9] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1986.

[10] IBM. Identity mixer, 2012.

[11] T. Iyama, S. Kiyomoto, K. Fukushima, T. Tanaka, and T. Takagi. *Efficient Implementation of Pairing on BREW Mobile Phones*, pages 326–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[12] K. Kluczniak, L. Hanzlik, and M. Kutyłowski. A formal concept of domain pseudonymous signatures. In *International Conference on Information Security Practice and Experience*, pages 238–254. Springer, 2016.

[13] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199. Springer, 1999.

[14] Microsoft. Microsoft U-Prove, 2015.

[15] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology—CRYPTO'92*, pages 31–53. Springer, 1992.

	Original Construction		Our Construction	
	Prover	Verifier	Prover	Verifier
$\mathbb{G}+$	0	0	$l+1$	$l+2$
$[\alpha]\mathbb{G}$	$2l+3$	0	$3l+5$	$l+3$
$\mathbb{G}_T\cdot$	$l+1$	$l+2$	0	0
$\mathbb{G}_T^{\alpha}$	$l+2$	$l+3$	0	0
$\mathbb{Z}_q+$	$l+2$	0	$l+2$	0
$\mathbb{Z}_q\cdot$	$l+3$	0	$l+3$	0
e	$l+2$	$5l+5$	0	$2l+3$
$\mathbb{G}$ sent	$2l+3$	0	$2l+4$	0
$\mathbb{G}_T$ sent	1	0	0	0
$\mathbb{Z}_q$ sent	$l+2$	1	$l+2$	1

Table 1: Comparison of our construction and the original protocol

APPENDIX

A. COMPARISON TO THE ORIGINAL PROTOCOLS

In order to stress the improvements of our construction from Section 3, we present Table 1 comparing the number of operations and communication costs for the protocols. The rows correspond to specific operations performed by the parties in columns. The numbers for the original construction are based upon the straightforward method using Okamoto-Schnorr Σ-protocol for a proof of exponents equality. We compare the number of:

1. elliptic points additions and scalar multiplications,
2. multiplications and exponentiations in $\mathbb{G}_T$,
3. exponents additions and multiplications,
4. bilinear pairings computations,
5. items of $\mathbb{G}$, $\mathbb{G}_T$ and exponents sent

with respect to the number of signature messages l (cf Figure 4).