An efficient self-blindable attribute-based credential scheme

Abstract: An attribute-based credential scheme allow a user to selectively show some of his properties (attributes), while keeping the others to himself. A number of such schemes exist, of which some additionally provide unlinkability: that is, if the same attributes were disclosed in two transactions, it is not possible to tell if one and the same or two different credentials were involved. Recently full-fledged implementation of such schemes on smart cards have emerged; however, these need to compromise the security level to achieve reasonable transaction speeds. In this paper we present a new unlinkable attribute-based credential scheme with a full security proof, using known hardness assumptions in the standard model. Defined on elliptic curves, the scheme involves bilinear pairings but only on the verifier's side, making it very efficient both in terms of speed and size. This should allow for fast and secure implementations on smart cards.

Keywords: Attribute-based credentials, unlinkability, bilinear pairings

1 Introduction

An attribute-based credential scheme allows a user, given a set of attributes $k_1, \dots, k_n$, to prove ownership of these attributes to a verifier, voluntarily disclosing some of them while keeping the others secret. A number of such credential schemes exist, of which some additionally provide *unlinkability*: that is, the verifier cannot tell whether two transactions did or did not originate from the same user (assuming the same attributes with the same values were disclosed in both transactions). This allows for very flexible identity management schemes, that are simultaneously very secure and privacy-friendly.

Two well-known schemes are Idemix [12, 31] and U-Prove [11, 39].[1] However, to date there is no provably secure scheme that is sufficiently efficient to allow truly safe implementations on smart cards, while also providing unlinkability of transactions. For example, since Idemix is based on the strong RSA-problem, one would want the keysize to be at least 2048 bits and preferably even 4096 bits; the IRMA project[2] has implemented Idemix on smart cards using 1024 bits. On the other hand, U-Prove is more efficient but does not provide unlinkability; in addition, its security is not fully proven, and it has even been suggested that its unforgeability is unprovable under standard intractability assumptions [4].

In this paper, we provide a new provably secure, efficient and unlinkable attribute-based credential scheme, that is based on the concept of *self-blindability* [43]: before showing the credential it is randomly modified into a new one (containing the same attributes) that is still valid. This results in a showing protocol which is such that the verifier learns nothing at all about the credential besides the fact that it is valid and the attributes that are disclosed (that is, the showing protocol is a zero-knowledge proof of knowledge). The scheme does not use the random oracle model (although usage of this model can lead to a performance increase through the Fiat-Shamir heuristic [23]), and it uses elliptic curves and bilinear pairings, allowing the same security level as RSA-type groups at much smaller key sizes. Although computing a pairing is a much more expensive operation than performing exponentiations on an elliptic curve, all pairings occur on the verifier's side. In addition, the kinds of pairing that we use (Type 3; see Definition 4) involves two distinct groups of which one is more expensive to do computations on, but the prover only needs to be aware of the cheaper of the two. These two facts ensure that the amount of work that the prover has to perform is minimal, and indeed our scheme is cheaper for the prover than any comparable scheme that we know of.

The unforgeability of our credential scheme will be implied by the LRSW assumption [13, 34, 35] introduced by Lysyanskaya et al., and used in many subsequent works (for example, [2, 13, 15, 45, 46]). Actually, for our purposes a weaker (in particular, non-interactive and thus falsifiable [38]) version of this assumption

1 We give a more detailed comparison of our scheme to those in the literature in Section 6.

2 https://www.irmacard.org

called the whLRSW assumption [46] will suffice. After having defined attribute-based credential schemes as well as unforgeability and unlinkability in the next section, we will discuss these assumptions in Section 3. In the same section we will introduce a signature scheme on the space of attributes, that will serve as the basis for our credential scheme. In section 4 we turn to our credential scheme, defining issuing and showing protocols, and proving that these provide unlinkability and unforgeability for our scheme. This in turn implies the unforgeability of the signature scheme. In Sections 5 and 6 we will discuss the performance of our scheme, and compare it to a number of other attribute-based credential schemes, in terms of features, efficiency and speed, and security.

2 Attribute-based credential schemes

First we fix some notation. We denote algorithms with calligraphic letters such as $\mathcal{A}$ and $\mathcal{B}$. By $y \leftarrow \mathcal{A}(x)$ we denote that y was obtained by running $\mathcal{A}$ on input x. If $\mathcal{A}$ is a deterministic algorithm then y is unique; if $\mathcal{A}$ is probabilistic then y is a random variable. We write $\mathcal{A}^O$ when algorithm $\mathcal{A}$ can make queries to oracle O. That is, A has an additional (read/write-once) on which it writes its queries; once it writes a special delimiter oracle O is invoked, and its answer appears on the query tape adjacent to the delimiter.

If $\mathcal{A}$ and $\mathcal{B}$ are interactive algorithms, we write $a \leftarrow \mathcal{A}(\cdot) \leftrightarrow \mathcal{B}(\cdot) \rightarrow b$ when $\mathcal{A}$ and $\mathcal{B}$ interact and afterwards output a and b, respectively. By $\mathcal{A} \xrightarrow{\blacksquare} \mathcal{B}$ we denote that algorithm $\mathcal{A}$ has black-box access to an interactive algorithm $\mathcal{B}$ – that is, $\mathcal{A}$ has oracle access to the next-message function function $\mathcal{B}_{x,y,r}(m)$ which, on input x that is common to $\mathcal{A}$ and $\mathcal{B}$, auxiliary input y and random tape r, specifies the message that $\mathcal{B}$ would send after receiving messages m. Finally, $|x|$ denotes the length of x in bits. For example, if x is an integer then $|x| = \lceil \log_2 x \rceil$.

For zero-knowledge proofs (see Section B) we will use the Camenisch-Stadler notation [14]. For example, if K, P_1, P_2 are elements of some (multiplicatively written) group then

$$\mathrm{PK}\big\{(k_1, k_2)\colon K = P_1^{k_1} P_2^{k_2}\big\}$$

denotes a zero-knowledge proof of knowledge of the numbers k_1, k_2 that satisfy the relation $K = P_1^{k_1} P_2^{k_2}$.

Definition 1. An attribute-based credential scheme consists at least of the following protocols.

KeyGen$(1^\ell, n)$ This algorithm takes as input a security parameter ℓ and the number of attributes n, and outputs the issuer's private key s and public key σ, which must contain the number n of attributes all credentials will have, and a description of the attribute space M.

Issue An interactive protocol between an issuer $\mathcal{I}$ and prover $\mathcal{P}$ that results in a credential c:

$$\mathcal{I}(\sigma, s, (k_1, \ldots, k_n)) \leftrightarrow \mathcal{P}(\sigma, k_0, (k_1, \ldots, k_n)) \rightarrow c.$$

Here k_0 is the prover's private key, that is to be chosen from the attribute space M by the prover; the Issue protocol should prevent the issuer from learning it. We assume that before execution of this protocol, the issuer and prover have reached agreement on the values of the attributes $k_1, \ldots, k_n$. The secret key and attributes $k_0, k_1, \ldots, k_n$ are contained in the credential c.[3]

ShowCredential An interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ which is such that, if c is a credential[4] issued using the Issue protocol over attributes $(k_1, \ldots, k_n)$ using private key s corresponding to public key σ, then for any disclosure set $\mathcal{D} \subset \{1, \ldots, n\}$ the prover can make the verifier accept:

$$\mathcal{P}(\sigma, c, \mathcal{D}) \leftrightarrow \mathcal{V}(\sigma, \mathcal{D}, (k_i)_{i \in \mathcal{D}}) \rightarrow 1.$$

Thus, the prover will have to notify the verifier in advance of the disclosure set $\mathcal{D}$ and disclosed attributes $(k_i)_{i \in \mathcal{D}}$.

We expect our attribute-based credential scheme to satisfy the following properties.

- *Unforgeability* (see Definition 2): Only the issuer can create valid credentials, and the attributes of valid credentials cannot be modified without invalidating the credential.
- *Multi-show unlinkability* (see Definition 3): If a verifier $\mathcal{V}$ participates in the ShowCredential protocol

3 We realize that in terms of terminology, it is rather unusual to include the secret key in the credential. We do this mainly for notational convenience in the later sections.

4 As in Idemix and U-Prove, our ShowCredential protocol can easily be extended to simultaneously show multiple credentials that have the same secret key (see Section 4.1), and to proving that the hidden attributes satisfy arbitrary linear combinations [11].

twice, in which the same attributes with the same values were disclosed, it should be impossible for it to tell whether both executions originated from the same credential or from two different ones.
- *Issuer unlinkability*: If in a run of the ShowCredential protocol certain attributes were disclosed, then of all credentials that the issuer issued with those attributes, the issuer cannot tell which one was used.
- *Non-repudiation*: the issuer cannot deny that the credential's signature was produced by it.
- *Offline issuer*: The issuer is not involved in the verification of credentials.
- *Selective disclosure*: Any subset of attributes contained in a credential can be disclosed.

We define unforgeability and both kinds of unlinkability of an attribute-based credential scheme in terms of the following games.

Definition 2 (unforgeability game). The unforgeability game of an attribute-based credential scheme between a challenger and an adversary $\mathcal{A}$ is defined as follows.

Setup For a given security parameter ℓ, the adversary decides on the number of attributes $n \geq 1$ that each credential will have, and sends n to the challenger. The challenger then runs the $\mathsf{KeyGen}(1^\ell, n)$ algorithm from the credential scheme and sends the resulting public key to the adversary.

Queries The adversary $\mathcal{A}$ can make the following queries to the challenger.

$\mathsf{Issue}(k_{1,j}, \ldots, k_{n,j})$ The challenger and adversary engage in the Issue protocol, with the adversary acting as the prover and the challenger acting as the issuer, over the attributes $(k_{1,j}, \ldots, k_{n,j})$. It may choose these adaptively.

$\mathsf{ShowCredential}(\mathcal{D}, k_1, \ldots, k_n)$ The challenger creates a credential with the specified attributes $k_1, \ldots, k_n$, and engages in the ShowCredential protocol with the adversary, acting as the prover and taking $\mathcal{D}$ as disclosure set, while the adversary acts as the verifier.

Challenge The challenger, now acting as the verifier, and the adversary, acting as the prover, engage in the ShowCredential protocol. If the adversary manages to make the verifier accept a credential with disclosed attributes $(k_i)_{i\in\mathcal{D}}$ (where $\mathcal{D} \neq \emptyset$), and there is no j such that $k_i = k_{i,j}$ for all $i \in \mathcal{D}$ (i.e., there is no single credential from one of the Issue queries containing all of the disclosed attributes $(k_i)_{i\in\mathcal{D}}$), then the adversary wins.

We say that the credential scheme is *unforgeable* if no probabilistic polynomial-time algorithm can win this game with non-negligible probability in the security parameter ℓ.

Next we turn to the unlinkability game. We define a single game that implies both multi-show and issuer unlinkability (see Appendix C).

Definition 3 (unlinkability game). The unlinkability game of an attribute-based credential scheme between a challenger and an adversary $\mathcal{A}$ is defined as follows.

Setup For a given security parameter ℓ, the adversary decides on the number of attributes $n \geq 1$ that each credential will have, and sends n to the challenger. The adversary then runs the $\mathsf{KeyGen}(1^\ell, n)$ algorithm from the credential scheme and sends the resulting public key to the challenger.

Queries The adversary $\mathcal{A}$ can make the following queries to the challenger.

$\mathsf{Issue}(k_{1,j}, \ldots, k_{n,j})$ The adversary chooses a set of attributes $(k_{1,j}, \ldots, k_{n,j})$, and sends these to the challenger. Then, acting as the issuer, the adversary engages in the Issue protocol with the challenger, issuing a credential j to the challenger having attributes $(k_{1,j}, \ldots, k_{n,j})$.

$\mathsf{ShowCredential}(j, \mathcal{D})$ The adversary and challenger engage in the showing protocol on credential j, the challenger acting as the prover and the adversary as the verifier. Each time the adversary may choose the disclosure set $\mathcal{D}$.

$\mathsf{Corrupt}(j)$ The challenger sends the entire internal state, including the secret key k_0, of credential j to the adversary.

Challenge The adversary chooses two uncorrupted credentials j_0, j_1 and a disclosure set $\mathcal{D} \subset \{1, \ldots, n\}$. These have to be such that the disclosed attributes from credential j_0 coincide with the ones from credential j_1, i.e., $k_{i,j_0} = k_{i,j_1}$ for each $i \in \mathcal{D}$. It sends the indices j_0, j_1 and $\mathcal{D}$ to the challenger, who checks that this holds; if it does not then the adversary loses.

Next, the challenger flips a bit $b \in_R \{0, 1\}$, and acting as the prover, it engages in the ShowCredential with the adversary on credential j_b. All attributes whose index is in $\mathcal{D}$ are disclosed.

Output The adversary outputs a bit b' and wins if $b = b'$.

We define the advantage of the adversary $\mathcal{A}$ as $\mathsf{Adv}_\mathcal{A} := |\Pr[b = b'] - 1/2|$. When no probabilistic polynomial-time algorithm can win this game with non-negligible advantage in the security parameter ℓ, then we say that the credential scheme is *unlinkable*.

The notion of unlinkability captures the idea that it is impossible for the verifier to distinguish two credentials from each other in two executions of the ShowCredential protocol, as long as they disclosed the same attributes with the same values. We will achieve this for our scheme by proving that our ShowCredential protocol is *black-box zero-knowledge* (see Definition 22), which essentially means that the verifier learns nothing at all besides the statement that the prover proves. Since the verifier learns nothing that it can use to link transactions, unlinkability follows from this (see Theorem 16).

3 Preliminaries

Definition 4. A bilinear group pair (G_1, G_2) consists of two cyclic groups (that we will write multiplicatively), both of prime order p, such that there exists a *bilinear map* or *pairing*; that is, a map $e\colon G_1 \times G_2 \to G_T$ (with G_T a multiplicative group of order p) satisfying the following properties:

- *Bilinearity*: For all $A, A' \in G_1$ and $B, B' \in G_2$ we have $e(AA', B) = e(A, B)e(A', B)$ and $e(A, BB') = e(A, B)e(A, B')$.
- *Non-degeneracy*: If $P \in G_1$, $Q \in G_2$ are two generators, then the element $e(P, Q)$ is a generator of G_T (that is, it is unequal to $1 \in G_T$).
- *Computability*: There exists an efficient algorithm for computing $e(A, B)$ for any $A \in G_1$, $B \in G_2$.

We only consider Type 3 pairings, that is, bilinear group pairs such that there is no efficiently computable isomorphism either from G_1 to G_2 or vice versa. Such pairings exist for particular types of elliptic curves; we mention for example [5, 36]. For more information about bilinear group pairs and pairings we refer to [24]; see also, for example, Chapters I and X from [9].

We will always use uppercase letters for elements of G_1 or G_2, while lowercase letters (including Greek letters) will be numbers, i.e., elements of $\mathbb{Z}_p$. We will always use the index i for attributes, and in the unforgeability proofs below we will use the index j for multiple provers or multiple credentials. For example, the number $k_{i,j}$ will refer to the i-th attribute of the credential of prover j. If a, b are two natural numbers with $a < b$, then we will sometimes for brevity write $[a, b]$ for the set $\{a, \ldots, b\}$.

We write $\nu(\ell) < \text{negl}(\ell)$ when the function $\nu\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligible; that is, for any polynomial p there exists an ℓ' such that $\nu(\ell) < 1/p(\ell)$ for all $\ell > \ell'$.

3.1 Intractability assumptions

The unforgeability of the credential and signature schemes defined in the paper will depend on the *whLRSW assumption* [46], which as we will show below, is implied by the LRSW assumption [34, 35] by Lysyanskaya et al. The latter assumption has been proven to hold in the generic group model [41], and has been used in a variety of schemes (for example, [2, 13, 15, 45, 46]). Although this assumption suffices to prove unforgeability of our scheme, it is stronger than we need. In particular, the LRSW assumption is an interactive assumption, in the sense that the adversary is given access to an oracle which it can use as it sees fit. We prefer to use the weaker whLRSW assumption, which is implied by the LRSW assumption but does not use such oracles. Consequentially, unlike the LRSW assumption itself, and like conventional hardness assumptions such as factoring and DDH, this assumption is falsifiable [38]. We describe both assumptions below; then we prove that the LRSW assumption implies the whLRSW assumption. After this we will exclusively use the latter assumption.

Let $e\colon G_1 \times G_2 \to G_T$ be a Type 3 pairing, where the order p of the three groups is ℓ bits, and let $a, z \in_R \mathbb{Z}_p^*$. If $(\kappa, K, S, T) \in \mathbb{Z}_p^* \times G_1^3$ is such that $S = K^a$ and $T = K^{z+\kappa az}$, then we call (κ, K, S, T) an *LRSW-instance.*

Definition 5 (LRSW assumption). Let e be as above, and let $O_{a,z}$ be an oracle that, when it gets $\kappa_j \in \mathbb{Z}_p^*$ as input on the j-th query, chooses a random $K_j \in_R G_1$ and outputs the LRSW-instance $(\kappa_j, K_j, K_j^a, K_j^{z+\kappa_j az})$. The *LRSW problem* is, when given $(p, e, G_1, G_2, G_T, Q, Q^a, Q^z)$ and oracle access to $O_{a,z}$, to output a new LRSW-instance $(\kappa, K, K^a, K^{z+\kappa az})$ where κ has never been queried to $O_{a,z}$. The *LRSW assumption* is that no probabilistic polynomial-time algorithm can solve the LRSW problem with non-negligible probability in ℓ. That is, for every probabilistic polynomial-time algorithm $\mathcal{A}$ we have

$$\begin{aligned}\Pr\Big[&a, z \in_R \mathbb{Z}_p^*;\ Q \in_R G_2 \setminus \{1\};\\ &\sigma \leftarrow (p, e, G_1, G_2, G_T, Q, Q^a, Q^z);\\ &(\kappa, K, S, T) \leftarrow \mathcal{A}^{O_{a,z}}(\sigma) :\\ &K \in G_1 \ \wedge\ \kappa \in \mathbb{Z}_p^* \ \wedge\ \kappa \notin Q\\ &\qquad \wedge\ S = K^a \ \wedge\ T = K^{z+\kappa az}\Big] < \text{negl}(\ell),\end{aligned}$$

where Q is the list of oracle queries sent to $O_{a,z}$, and where the probability is over the choice of a, z, Q, and the randomness used by $\mathcal{A}$ and the oracle $O_{a,z}$.

Definition 6 (q-whLRSW assumption). Let e be as above, and let $\{(\kappa_j, K_j, K_j^a, K_j^{z+\kappa_j az})\}_{j=1,\ldots,q}$ be a list of q LRSW-instances, where the κ_j and K_j are randomly distributed in $\mathbb{Z}_p^*$ and G_1, respectively. The *q-whLRSW problem* (for q-wholesale LRSW [46]) is, when given this list along with $(p, e, G_1, G_2, G_T, Q, Q^a, Q^z)$, to output a new LRSW-instance $(\kappa, K, K^a, K^{z+\kappa az})$ where $\kappa \notin \{\kappa_1, \ldots, \kappa_q\}$. The *$q$-whLRSW assumption* is that no probabilistic polynomial-time algorithm can solve the q-whLRSW problem with non-negligible probability in ℓ. That is, for every probabilistic polynomial-time algorithm $\mathcal{A}$ we have

$$\begin{aligned}\Pr\Big[&a, z, \kappa_1, \ldots, \kappa_q \in_R \mathbb{Z}_p^*;\ K_1, \ldots, K_q \in_R G_1 \setminus \{1\};\\ &Q \in_R G_2 \setminus \{1\};\ \sigma \leftarrow (p, e, G_1, G_2, G_T, Q, Q^a, Q^z);\\ &(\kappa, K, S, T) \leftarrow \mathcal{A}(\sigma, \{\kappa_j, K_j, K_j^a, K_j^{z+\kappa_j az}\}_j) :\\ &K \in G_1 \ \wedge\ \kappa \in \mathbb{Z}_p^* \ \wedge\ \kappa \notin \{\kappa_1, \ldots, \kappa_q\}\\ &\qquad \wedge\ S = K^a \ \wedge\ T = K^{z+\kappa az}\Big] < \operatorname{negl}(\ell), \quad (1)\end{aligned}$$

where the probability is over the choice of a, z, $\kappa_1, \ldots, \kappa_q$, $K_1, \ldots, K_q, Q$, and the randomness used by $\mathcal{A}$.

Finally we define an unparameterized version of the assumption above by allowing q to be polynomial in ℓ, in the following standard way (e.g., [10]). Intuitively, the reason that this unparameterized assumption is implied by the LRSW assumption is simple: if there is no adversary that can create LRSW-instances when it can (using the oracle) control the κ's of the LRSW-instances that it gets as input, then an adversary that can create them *without* having control over the κ's also cannot exist.

Definition 7. Let e, p and $\ell = |p|$ be as above. The *whLRSW assumption* states that for all polynomials $q\colon \mathbb{N} \to \mathbb{N}$, the $q(\ell)$-whLRSW assumption holds.

Proposition 8. *The LRSW assumption implies the whLRSW assumption.*

Proof. Suppose that the whLRSW assumption does not hold, i.e., there is a polynomial q and a probabilistic polynomial-time algorithm $\mathcal{A}$ such that if $\mathcal{A}$ is given a list of $q(\ell)$ LRSW-instances, then it can produce a new valid LRSW-instance with non-negligible probability in ℓ. Now we create an algorithm $\mathcal{B}$ that violates the LRSW assumption. Algorithm $\mathcal{B}$ is given $\sigma = (p, e, G_1, G_2, G_T, Q, Q^a, Q^z)$ and oracle access to $Q_{a,z}$, and it operates as follows.

– It randomly chooses $q(\ell)$ values $\kappa_j \in_R \mathbb{Z}_p^*$;
– For each j, $\mathcal{B}$ calls $O_{a,z}(\kappa_j)$, obtaining $q(\ell)$ valid LRSW-instances $L_j = (\kappa_j, K_j, K_j^a, K_j^{z+\kappa_j az})$;
– Algorithm $\mathcal{B}$ runs and returns the output of

$$\mathcal{A}(\sigma, L_1, \ldots, L_{q(\ell)}).$$

Then $\mathcal{B}$ is a probabilistic polynomial-time algorithm whose success probability is the same as that of $\mathcal{A}$, which is non-negligible by assumption. This contradicts the LRSW assumption. □

Thus if we prove that our scheme is safe under the whLRSW asssumption, then it is also safe under the LRSW assumption. Additionally, we have found that the whLRSW assumption can be proven by taking an extension [8] of the Known Exponent Assumption [20], so that unforgeability of our scheme can also be proven by using this assumption. However, because of space restrictions this proof is not included here.

3.2 A signature scheme on the space of attributes

In this section we introduce a signature scheme on the space of attributes. This signature scheme will be the basis for our credential scheme, in the following sense: the Issue protocol that we present in the section after this will enable issuing such signatures over a set of attributes to users, while the ShowCredential protocol allows the user to prove that it has a signature over any subset of its signed attributes.

The Chaum-Pedersen signature scheme [17] serves as the basis for our signature scheme on the attribute space, and works as follows. Let $e\colon G_1 \times G_2 \to G_T$ be a bilinear group pair of Type 3, and let $Q \in G_2$ be a generator. The issuer's private and public keys are $a \in \mathbb{Z}_p$ and (e, A, Q) respectively, where $A = Q^a$. A signature on a message $K \in G_1$ is $S = K^a$, and it is verified by $e(S, Q) \overset{?}{=} e(K, A)$.

There are two immediate issues with this signature scheme. First, if (K, S) is a valid message-signature pair and $c \in \mathbb{Z}_p$, then (K^c, S^c) is also valid. Second, if (K_1, S_1) and (K_2, S_2) are both valid then (K_1K_2, S_1S_2) is also valid. That is, given valid message-signature pairs we can construct valid signatures on arbitrary powers and arbitrary products of the messages, without knowing the secret key. This signature scheme, then, is definitely not existentially unforgeable. On the other hand, this ability to modify valid signatures into new ones will allow us to create a showing protocol for credentials that is zero-knowledge in Section 4.

Definition 9 (Signature scheme on attribute space). The signature scheme is as follows.

$\mathsf{KeyGen}(1^\ell, n)$ The issuer generates a Type 3 pairing $e\colon G_1 \times G_2 \to G_T$, such that $|p| = \ell$ where p is the prime order of the three groups. Next it takes a generator $Q \in_R G_2$, and numbers $a, a_0, \dots, a_n, z \in_R \mathbb{Z}_p^*$ and sets $A = Q^a, A_0 = Q^{a_0}, \dots, A_n = Q^{a_n}$, and $Z = Q^z$. The public key is the tuple $\sigma = (p, e, Q, A, A_0, \dots, A_n, Z)$ and the private key is the tuple $(a, a_0, \dots, a_n, z)$.

$\mathsf{Sign}(k_0, \dots, k_n)$ The issuer chooses $\kappa \in_R \mathbb{Z}_p^*$ and $K \in_R G_1$, and sets $S = K^a, S_0 = K^{a_0}, \dots, S_n = K^{a_n}$, and $T = (KS^\kappa \prod_{i=0}^n S_i^{k_i})^z$. The signature is $(\kappa, K, S, S_0, \dots, S_n, T)$.

$\mathsf{Verify}((k_0, \dots, k_n), (\kappa, K, S, S_0, \dots, S_n, T), \sigma)$ The signature is checked by setting $C = KS^\kappa \prod_{i=0}^n S_i^{k_i}$ and verifying that $K, C \neq 1$, as well as

$$e(T, Q) \overset{?}{=} e(C, Z), \qquad e(S, Q) \overset{?}{=} e(K, A),$$
$$e(S_i, Q) \overset{?}{=} e(K, A_i) \qquad \text{for each } i = 0, \dots, n.$$

The numbers $k_n \in \mathbb{Z}_p$ are the attributes. Although p may vary each time the $\mathsf{KeyGen}(1^\ell, n)$ algorithm is invoked on a fixed security parameter ℓ, the attribute space $\mathbb{Z}_p$ will always contain $\{0, \dots, 2^{\ell-1}\}$. In our credential scheme in section 4, the zeroth attribute k_0 will serve as the prover's secret key, but at this point it does not yet have a special role.

Although the element $C = KS^\kappa \prod_{i=0}^n S_i^{k_i}$ is, strictly speaking, not part of the signature and therefore also not part of the credential (since it may be calculated from κ, the attributes $(k_0, \dots, k_n)$ and the elements $(K, S, S_0, \dots, S_n)$), we will often think of it as if it is. Finally, we call a message-signature pair, i.e., a tuple of the form $((k_0, \dots, k_n), (\kappa, K, S, S_0, \dots, S_n, T))$ where $(\kappa, K, S, S_0, \dots, S_n, T)$ is a valid signature over $(k_0, \dots, k_n)$, a *credential.*

Notice that if $(k_0, \dots, k_n), (\kappa, K, S, S_0, \dots, S_n, T)$ is a valid credential, then $(k_0, \dots, k_n), (\kappa, K^\alpha, S^\alpha, S_0^\alpha, \dots, S_n^\alpha, T^\alpha)$ for any $\alpha \in \mathbb{Z}_p^*$ is another valid credential having the same attributes. That is, in the terminology of Verheul [43] our credentials are *self-blindable.* At the same time, it is not at all clear how to change the credential into another one over different attributes, and indeed, we essentially prove in Theorem 14 that this is not possible. This self-blindability is what makes this signature scheme suitable for the purpose of creating an unlinkable $\mathsf{ShowCredential}$ protocol.

The number κ will play a critical role in the unforgeability proof of our signature and credential schemes (Theorem 14).[5]

Once the issuer has chosen an amount of attributes n and computed his public key, he can easily enlarge the maximum amount n of attributes that the scheme allows by generating additional secret keys $a_{n+1}, \dots, a_{n'} \in_R \mathbb{Z}_p^*$, and adding the elements $A_{n+1} = Q^{a_{n+1}}, \dots, A_{n'} = Q^{a_{n'}}$ to his public key. All credentials that have been computed beforehand will remain valid under this modified public key, and their value for the new attributes $k_{n+1}, \dots, k_{n'}$ will be 0.

Theorem 10. *Our credentials are existentially unforgeable under adaptively chosen message attacks, under the whLRSW assumption.*

We will prove this after we have proven unforgeability of our credential scheme (Theorem 14).

4 The credential scheme

In this section we present our credential scheme. The strategy is as follows: having defined an unforgeable signature scheme on the set of attributes $\mathbb{Z}_p^n$ (Definition 9), we provide an issuing protocol, in which the issuer grants a credential to a prover, and a showing protocol, which allows a prover to give a zero-knowledge proof to a verifier that he possesses a credential, revealing some of the attributes contained in the credential while keeping the others secret. The Issue protocol[6] is shown in Figure 1, and the $\mathsf{ShowCredential}$ protocol is shown in Figure 2. Here and in the remainder of the paper, we will write $\mathcal{D} \subset \{1, \dots, n\}$ for the index set of the disclosed attributes, and

$$\mathcal{C} = \{1, \dots, n\} \setminus \mathcal{D}$$

5 In the Idemix credential scheme [12], the role of the number v, as well as the way in which it is computed in the Issue protocols, is similar to κ. We could have eased the notation somewhat by denoting it as an extra attribute k_{n+1}, but because it plays a rather different role than the other attributes (it is part of the signature), we believe this would create more confusion than ease.

6 Although in the Issue protocol one could say that it is the issuer that proves something (namely that it knows the secret key of the credential scheme), we will for consistency with the rest of the paper here too refer to the user as the prover.

for the index set of the undisclosed attributes. The secret key k_0 is not part of this set, as it is always kept secret. Thus, $n = 1 + |\mathcal{C}| + |\mathcal{D}|$.

The Issue protocol is such that both parties contribute to κ and K with neither party being able to choose the outcome in advance. This is important, because only then does Theorem 14 guarantee unforgeability. Additionally, the issuer is prevented from learning the values of κ and the secret key k_0.

As noted earlier, we assume that the prover and issuer have agreed on the attributes $k_1, \dots, k_n$ to be contained in the credential before executing this protocol. Similarly, we assume that the prover sends the dislosure set $\mathcal{D}$ and disclosed attributes $(k_i)_{i\in\mathcal{D}}$ to the verifier prior to executing the ShowCredential protocol.

Remark 11. Although we have not explicitly denoted so in the protocols, it is important (in both protocols) that whenever one party receives a group element (say, $X = (x, y)$ from G_1 or G_2) from the other party, that it checks that X is indeed a valid element from G_1 or G_2. That is, it should be verified that the coordinates (x, y) of X satisfy the equation that defines the curve. For example, if the curve is defined using a Weierstrass equation with coefficients a, b, then

$$y^2 = x^3 + ax + b$$

must hold, in order to prevent invalid curve attacks such as for example those from [1]. For the same reason, if the group G_1 or G_2 is a proper subgroup of the curve, then it must also be checked that the element is indeed a member of this subgroup.

Mathematically, we can formalize what the ShowCredential protocol should do as follows. The common knowledge of the prover and verifier when running the ShowCredential protocol consists of elements of the following formal language:

$$L = \left\{ \left(\sigma, \mathcal{D}, (k_i)_{i\in\mathcal{D}}\right) \mid \mathcal{D} \subset \{1, \dots, n\},\ k_i \in \mathbb{Z}_p \,\forall\, i \in \mathcal{D} \right\} \tag{2}$$

where σ ranges over the set of public keys of the credential scheme. In addition, let the relation R be such that $R(x, w) = 1$ only if $x = (\sigma, \mathcal{D}, (k_i)_{i\in\mathcal{D}}) \in L$, and $w = ((k'_0, \dots, k'_n), s)$ is a valid credential with respect to σ, with $k'_i = k_i$ for $i \in \mathcal{D}$ (i.e., the disclosed attributes $(k_i)_{i\in\mathcal{D}}$ are contained in the credential w.) Thus the equation $R(x, w) = 1$ holds only if w is a valid credential having attributes $(k_i)_{i\in\mathcal{D}}$.

In Theorems 12, 13 and 15 we will prove that our ShowCredential protocol is a black-box zero-knowledge proof of knowledge (see Definition 22) for this relation. From this fact unforgeability and unlinkability will follow.

Theorem 12. *The showing protocol is complete with respect to the language L: if a prover has a valid credential then it can make the verifier accept.*

Proof. If the prover follows the ShowCredential protocol, then $e(\bar{K}, A) = e(K^\alpha, Q^a) = e(K^{\alpha a}, Q) = e(S^\alpha, Q) = e(\bar{S}, Q)$, so the first verification that the verifier does will pass. An almost identical calculation shows that the second and third verifications pass as well. As to the proof of knowledge, setting $\bar{C} = C^\alpha$ we have

$$\begin{aligned} \tilde{C}^\beta \bar{S}^\kappa \bar{S}_0^{k_0} \prod_{i\in\mathcal{C}} \bar{S}_i^{k_i} &= \bar{C}^{-1} \bar{S}^\kappa \bar{S}_0^{k_0} \prod_{i\in\mathcal{C}} \bar{S}_i^{k_i} \\ &= \bar{K}^{-1} \prod_{i\in\mathcal{D}} \bar{S}_i^{-k_i} = D, \end{aligned} \tag{3}$$

so the prover can perform this proof without problem. □

4.1 Combining credentials using the private key

Let a bilinear pairing $e\colon G_1 \times G_2 \to G_T$ be fixed, and let σ and σ' be two public keys defined in these groups (not necessarily distinct). In this scenario multiple credentials can be bound together using their private keys k_0, as follows.

Let w and w' be two credentials valid with respect to σ and σ', such that w and w' have the same secret key:

$$\begin{aligned} w &= ((k_0, k_1, \dots, k_n), (K, S, S_0, \dots, S_n, T)), \\ w' &= ((k_0, k'_1, \dots, k'_n), (K', S', S'_0, \dots, S'_n, T')), \end{aligned}$$

Now the prover performs the ShowCredential credential once for each credential, but replaces the two proofs of knowledge by the following combined proof of knowledge:

$$\begin{aligned} \mathrm{PK}\Big\{ &\left(k_0, \beta, \beta', \kappa, \kappa', (k_i)_{i\in\mathcal{C}}, (k'_j)_{j\in\mathcal{C}'}\right) : \\ &D = \tilde{C}^\beta \bar{S}^\kappa \bar{S}_0^{k_0} \prod_{i\in\mathcal{C}} \bar{S}_i^{k_i} \\ &\wedge D' = (\tilde{C}')^{\beta'} (\bar{S}')^{\kappa'} (\bar{S}'_0)^{k_0} \prod_{i\in\mathcal{C}} (\bar{S}'_i)^{k'_i} \Big\} \end{aligned}$$

Common information: Attributes $k_1, \dots, k_n$, issuer's public key $\sigma = (p, e, Q, A, A_0, \dots, A_n, Z)$

Prover		**Issuer**
knows secret key k_0		knows $a, a_0, \dots, a_n$, z
	$\longleftarrow$	send $\bar{K} \in_R G_1, \bar{S} = \bar{K}^a, \bar{S}_0 = \bar{K}^{a_0}$
choose $\alpha, \kappa' \in_R \mathbb{Z}_p^*$		
set $K = \bar{K}^\alpha, S = \bar{S}^\alpha, S_0 = \bar{S}_0^\alpha$		
send $K, S, S_0, R = S^{\kappa'} S_0^{k_0}$	$\longrightarrow$	
		verify $K \neq \bar{K}$, $K = S^{1/a} = S_0^{1/a_0}$
$\mathrm{PK}\{(\kappa', k_0) \colon R = S^{\kappa'} S_0^{k_0}\}$	$\longleftrightarrow$	
		random $\kappa'' \in_R \mathbb{Z}_p$
		set $S_i = K^{a_i}$ $\forall i \in [1, n]$
		set $T = \left(K S^{\kappa''} R \prod_{i=1}^n S_i^{k_i}\right)^z$
	$\longleftarrow$	send $S_1, \dots, S_n, T, \kappa''$
set $\kappa = \kappa' + \kappa''$		
$\mathsf{Verify}((k_0, \dots, k_n), (\kappa, K, S, S_0, \dots, S_n, T), \sigma)$		
return $(k_0, \dots, k_n), (\kappa, K, S, S_0, \dots, S_n, T)$		

Fig. 1. The Issue protocol.

Common information: Issuer's public key $\sigma = (p, e, Q, A, A_0, \dots, A_n, Z)$; disclosure set $\mathcal{D}$, undisclosed set $\mathcal{C} = \{1, \dots, n\} \setminus \mathcal{D}$; disclosed attributes $(k_i)_{i \in \mathcal{D}}$

Prover		**Verifier**
knows $K, S, S_0, \dots, S_n, \kappa, (k_i)_{i \in \mathcal{C}}, C, T$		
choose $\alpha, \beta \in_R \mathbb{Z}_p^*$		
set $\bar{K} = K^\alpha$, $\bar{S} = S^\alpha$, $\bar{S}_i = S_i^\alpha$ $\forall i \in [0, n]$		
set $\tilde{C} = C^{-\alpha/\beta}$, $\tilde{T} = T^{-\alpha/\beta}$		
send $\bar{K}$, $\bar{S}$, $(\bar{S}_i)_{i=0,\dots,n}$, $\tilde{C}$, $\tilde{T}$	$\longrightarrow$	
set $D = \bar{K}^{-1} \prod_{i \in \mathcal{D}} \bar{S}_i^{-k_i}$		set $D = \bar{K}^{-1} \prod_{i \in \mathcal{D}} \bar{S}_i^{-k_i}$
$\mathrm{PK}\{(\beta, \kappa, k_0, k_i)_{i \in \mathcal{C}} \colon D = \tilde{C}^\beta \bar{S}^\kappa \bar{S}_0^{k_0} \prod_{i \in \mathcal{C}} \bar{S}_i^{k_i}\}$	$\longleftrightarrow$	
		verify $e(\bar{K}, A) \stackrel{?}{=} e(\bar{S}, Q)$
		and $e(\bar{K}, A_i) \stackrel{?}{=} e(\bar{S}_i, Q)$ $\forall i \in [0, n]$
		and $e(\tilde{C}, Z) \stackrel{?}{=} e(\tilde{T}, Q)$

Fig. 2. The ShowCredential protocol.

This proves the same as the two separate proofs of knowledge, as well as the fact that the two credentials have the same secret key k_0.

This assumes that the prover can control the private key during the issuing process, and this is indeed the case (see Figure 1). To ensure that each prover has its own distinct secret key k_0, the issuing protocol could be modified as follows. In one version, the issuer also chooses a random $k_0'' \in_R \mathbb{Z}_p^*$, and then sets and sends

$$T = \left(K S^{\kappa''} S_0^{k_0''} R \prod_{i=1}^n S_k^{k_i}\right)^z$$

to the prover together with κ'' and k_0''. The secret key of the new credential will then be not k_0 but $k_0 + k_0''$. In this way neither party can control the outcome of the secret key, and the issuer is prevented from learning its final value.

In the other version, the following happens:

- The prover and issuer together decide in advance on a public key σ (that may or may not equal the issuer's public key), and a (possibly empty) disclosure set $\mathcal{D}$ and set of attributes $(k_i)_{i \in \mathcal{D}}$).
- The prover first performs the ShowCredential protocol, with the issuer acting as verifier, on a credential w that is valid with respect to σ, disclosing the attributes $(k_i)_{i \in \mathcal{D}}$. If successful, the prover and issuer perform the Issue protocol as in Figure 1. However, the prover combines the proof of knowledge over the hidden attributes with the proof of knowledge over k_0 and κ in the Issue protocol, additionally show-

ing that k_0 coincides with the secret key from the credential that it showed.

4.2 Unforgeability

Lemma 13. *With respect to the language L defined in* (2)*, the* ShowCredential *protocol is black-box extractable in the sense of Definition 22.*

Proof. By Definition 22, we must show the existence of an extractor χ satisfying the following: for all $x \in L$, if a probabilistic polynomial-time algorithm $\mathcal{P}^*$, acting as the prover, can successfully run the showing protocol with a verifier with probability $\epsilon(|x|)$, then there is an algorithm χ that, when given black-box access to $\mathcal{P}^*$, extracts with probability $\epsilon(|x|)-\nu(|x|)$ a valid credential from $\mathcal{P}^*$, where $\nu(|x|)$ is some negligible function.

The extractor χ acts as follows:

– it stores the group elements $\bar{K}, \bar{S}, (\bar{S}_i)_{i=0,\ldots,n}, \tilde{T}$ that the prover sends to it;
– it extracts the numbers β, κ, k_0 and $(k_i)_{i\in\mathcal{D}}$ from the proof of knowledge;
– it sets $\bar{T} = \tilde{T}^{-\beta}$; note that if T was valid then we have $\bar{T} = T^{\alpha} = C^{\alpha z} = (\bar{K}\bar{S}^{\kappa}\bar{S}_0^{k_0}\prod_{i=0}^{n}\bar{S}_i^{k_i})^z$,
– it returns $(k_0, \ldots, k_n), (\kappa, \bar{K}, \bar{S}, \bar{S}_0, \ldots, \bar{S}_n, \bar{T})$.

The only action that χ performs that normal verifiers cannot perform is the extraction of the numbers $\beta, \kappa, k_0, (k_i)_{i\in\mathcal{C}}$ from the proof of knowledge. Therefore, if the prover $\mathcal{P}^*$ convinces normal verifiers with probability $\epsilon(|x|)$, then χ will succeed with probability $\epsilon(|x|) - \nu_D(|x|)$, where $\nu_D(|x|)$ is the decrease in probability associated to the extractor for the proof of knowledge of $\beta, \kappa, k_0, (k_i)_{i\in\mathcal{C}}$. Since ν_D is negligible by Definition 22, this proves the claim. □

In the proof of the unforgeability theorem, we will need a tuple $(\hat{K}, \hat{S}, \hat{S}_0, \ldots, \hat{S}_n) \in G_1^{n+3}$ such that $\hat{S} = \hat{K}^a$ and $\hat{S}_i = \hat{K}^{a_i}$ for all i. For that reason we will henceforth assume that such a tuple is included in the issuer's public key (we will need this tuple also when proving unlinkability of our scheme in Theorem 15).[7] Notice that the presence of this tuple does not endow the adversary with any extra capabilities, as it could already obtain such a tuple in an Issue query to the challenger.

Theorem 14. *Our credential scheme is unforgeable under the whLRSW assumption.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ that can break unforgeability of our scheme, using q_I Issue queries and q_S ShowCredential queries. Then we build an algorithm $\mathcal{B}$ that contradicts the q-whLRSW assumption with $q = q_I + q_S$. As q must be polynomial in ℓ (otherwise adversary $\mathcal{A}$ could not be polynomial-time), this in turn contradicts the whLRSW assumption.

Consider a setup without attributes, i.e., $n = 0$. The public key of this setup is $Q, \bar{A} = Q^{\bar{a}}, Z = Q^z$ (where the reason for the bar on top of the last group element will become clear shortly), and a valid credential would be $\kappa, K, \bar{S} = K^{\bar{a}}, T = (K\bar{S}^{\bar{\kappa}})^z = K^{z+\bar{\kappa}\bar{a}z}$; i.e., a LRSW-instance. We will refer to this setup as the original setup. Suppose that algorithm $\mathcal{B}$ is given a list of $q = q_I + q_S$ credentials

$$\kappa_j, K_j, \bar{S}_j, T_j = (K\bar{S}_j^{\bar{\kappa}_j})^z$$

that are valid with respect to this setup, along with $Q, \bar{A}, Z$. Algorithm $\mathcal{B}$, acting as the challenger, engages in the unforgeability game with $\mathcal{A}$, acting as follows.

Setup It lets $\mathcal{A}$ decide on the number of attributes n. Next it generates $a, a_0, \ldots, a_n \in_R \mathbb{Z}_p^*$ and then sets $A = \bar{A}^a$ and $A_i = \bar{A}^{a_i}$ for $i = 0, \ldots, n$. Then it sends the new setup

$$Q, Z, A, A_0, \ldots, A_n$$

to $\mathcal{A}$.

Queries Challenger $\mathcal{B}$ answers queries as follows. We use the index j here for both kinds of queries; i.e., if either one of the queries is performed then j is raised by 1.

Issue$(k_{1,j}, \ldots, k_{n,j})$: Challenger $\mathcal{B}$ engages in the issuing protocol with $\mathcal{A}$ on the specified attributes. Using credential $\kappa_j, K_j, \bar{S}_j, T_j$, note that the challenger can compute elements $S_{i,j}$ which are valid with respect to the new setup by

$$S_j = \bar{S}_j^a \qquad \text{and} \qquad S_{i,j} = \bar{S}_j^{a_i}$$

for all $j = 0, \ldots, m$. Using these elements it performs the issuing protocol with respect to the new setup normally, but when the adversary proves knowledge of its private key $k_{0,j}$ and the number κ', $\mathcal{B}$ extracts

7 Note that credential owners already have such a tuple; verifiers can obtain one simply by executing the ShowCredential protocol; and issuers can of course create such tuples by themselves. Therefore in practice, each party participating in the scheme will probably already have such a tuple, so that including it in the public key may not be necessary in implementations.

these numbers from the proof of knowledge. Next, it solves the equation

$$\kappa_j a + \sum_{i=0}^{n} k_{i,j} a_i = \bar{\kappa}_j \tag{4}$$

to κ_j (i.e., it sets $\kappa_j = (\sum_{i=0}^{n} k_{i,j} a_i - \bar{\kappa}_j)/a$), and uses the number $\kappa_j - \kappa'$ as the value for κ'' in the remainder of the protocol.

Notice that we then have

$$\begin{aligned} K_j \bar{S}_j^{\bar{\kappa}_j} &= K_j \bar{S}_j^{\kappa_j a + \sum_{i=0}^{n} k_{i,j} a_i} \\ &= K_j S_j^{\kappa_j} \prod_{i=0}^{n} S_{i,j}^{k_{i,j}}, \end{aligned}$$

and

$$\begin{aligned} e(S_{i,j}, Q) &= e(\bar{S}_j^{a_i}, Q) = e(K_j^{\bar{a} a_i}, Q) \\ &= e(K_j, \bar{A}^{a_i}) = e(K_j, A_i), \end{aligned}$$

and similarly $e(S_j, Q) = e(K_j, A)$. This means that $(k_{0,j}, \ldots, k_{n,j}), (\kappa_j, K_j, S_j, S_{0,j}, \ldots, S_{n,j}, T_j)$ is a valid credential with respect to the new setup.

ShowCredential$(\mathcal{D}, k_{1,j}, \ldots, k_{n,j})$: The challenger $\mathcal{B}$ embeds the attributes $k_{1,j}, \ldots, k_{n,j}$ in one of its own credentials as it does in Issue queries, choosing a κ_j and $k_{0,j}$ itself. Next it performs the ShowCredential protocol with the adversary over $(k_{i,j})_{i \in \mathcal{D}}$ as requested.

Output When $\mathcal{A}$ and $\mathcal{B}$ engage in the ShowCredential protocol, $\mathcal{B}$ uses the extractor guaranteed to exist by Lemma 13 above, obtaining a credential

$$(k_0, \ldots, k_n), (\kappa, K, S, S_0, \ldots, S_n, T).$$

Then $\mathcal{B}$ calculates $\bar{\kappa} = \kappa a + \sum_{i=0}^{n} k_i a_i$ and outputs $\bar{\kappa}, K, \bar{S} = S^{1/a}, T$.

If the output of $\mathcal{A}$ is a valid credential with respect to the new setup, then $S = K^{\bar{a} a} = \bar{S}^a$ and $S_i = K^{\bar{a} a_i} = \bar{S}^{a_i}$. Also, setting $\bar{\kappa} = \kappa a + \sum_{i=0}^{n} k_i a_i$, T equals

$$\begin{aligned} T &= \left(K S^{\kappa} \prod_{i=0}^{n} S_i^{k_i} \right)^z = \left(K \bar{S}^{\kappa a + \sum_{i=0}^{n} k_i a_i} \right)^z \\ &= \left(K \bar{S}^{\bar{\kappa}} \right)^z. \end{aligned}$$

This implies that the credential $\bar{\kappa}$, K, $\bar{S}$, T is valid with respect to the original setup. In order to derive a contradiction with the q-whLRSW assumption, we now show that with overwhelming probability $\bar{\kappa} \neq \bar{\kappa}_j$ for all j.

Suppose that there is a non-negligible chance that adversary $\mathcal{A}$ wins such that $\bar{\kappa} = \bar{\kappa}_j$ for some j. Then we have for this j

$$S^{\kappa} \prod_{i=0}^{n} S_i^{k_i} = \bar{S}^{\bar{\kappa}} = \bar{S}^{\bar{\kappa}_j} = S^{\kappa_j} \prod_{i=0}^{n} S_j^{k_{i,j}},$$

or equivalently,

$$1 = S^{\kappa - \kappa_j} \prod_{i=0}^{n} S_i^{k_i - k_{i,j}}. \tag{5}$$

Now there are two possibilities: either j corresponds to an Issue query or to a ShowCredential query. If j was an Issue query, then it follows from the fact that $\mathcal{A}$ won that the DL-representation (5) of 1 above is nontrivial. On the other hand, if j was a ShowCredential query, then the value of κ and k_0 that challenger $\mathcal{B}$ chose and used are information-theoretically hidden from $\mathcal{A}$. This means that the probability that $\mathcal{A}$ chose the values of κ, k_i exactly such that $\kappa = \kappa_j$ and $k_i = k_{i,j}$ is negligible, so that the DL-representation above will still be nontrivial with overwhelming probability. In both cases the DL-representation is thus nontrivial. Now, since the elements $\hat{S}, \hat{S}_0, \ldots, \hat{S}_n$ have the same relative discrete logarithms as the elements $S, S_0, \ldots, S_n$, this results in the following non-trivial DL-representation:

$$1 = \hat{S}^{\kappa - \kappa_j} \prod_{i=0}^{n} \hat{S}_i^{k_i - k_{i,j}}. \tag{6}$$

Notice that it is easy to check if this holds for some j even without knowledge of the secret key $a, a_0, \ldots, a_n, z$. We can therefore exploit this ability of the adversary without knowledge of these numbers to contradict Proposition 21 as follows. Let $P, X_0, \ldots, X_{n+1} \in G_1$, $Q, Y_0, \ldots, Y_{n+1} \in G_2$ be given, with $e(P, Y_i) = e(X_i, Q)$. We construct a non-trivial DL-representation of 1 with respect to $X_1, \ldots, X_n$ as follows.

– Set $\hat{K} = P, \hat{S} = X_0, \hat{S}_i = X_{i+1}$, and $A = Y_0, A_i = Y_{i+1}$. Take $z \in_R \mathbb{Z}_p^*$ and set $Z = Q^z$.
– Play the unforgeability game with the adversary with respect to this public key. In each query, generate a new valid tuple $(K, S, S_0, \ldots, S_n)$ by taking a random number $r \in \mathbb{Z}_p^*$ and setting $K = \hat{K}^r$, $S = \hat{S}^r$, and $S_i = \hat{S}_i^r$ for $i = 0, \ldots, n$. At the end of the game, return the resulting DL-representation (6) of 1.

As this would contradict Proposition 21, we conclude that we must have $\bar{\kappa} \neq \bar{\kappa}_j$ with overwhelming probability. But then the output $(\bar{\kappa}, K, \bar{S}, T)$ of algorithm $\mathcal{B}$ would be a new valid LRSW-instance, contradicting the q-whLRSW assumption. □

The unforgeability of the credential scheme implies that of the underlying signature scheme, as follows.

Proof of Theorem 10. Suppose that there exists an adversary $\mathcal{A}$ that can forge the signature scheme from Section 3. Then we create a forger $\mathcal{B}$ for our credential scheme as follows. Forger $\mathcal{B}$ is the challenger of adversary $\mathcal{A}$ in the signature scheme unforgeability game, and the adversary in the credential scheme unforgeability game. It operates as follows.

Setup Forger $\mathcal{B}$ receives a public key from its challenger and forwards it to the adversary $\mathcal{A}$.

Queries Whenever adversary $\mathcal{A}$ requests a signature on a set of attributes $(k_0, \ldots, k_n)$, $\mathcal{B}$ performs an Issue query on these attributes with its challenger. It sends the resulting signature to $\mathcal{A}$.

Output If $\mathcal{A}$ outputs a valid new credential then $\mathcal{B}$ uses this in the ShowCredential protocol with its challenger.

Then the success probability of $\mathcal{B}$ will be the same as that of $\mathcal{A}$. □

4.3 Anonymity

In order to prove that our ShowCredential protocol is zero-knowledge with respect to the language L from equation (2), we must show the existence of a simulator whose behavior is from the perspective of a verifier indistinguishable from an honest prover. In order to achieve this, the simulator will need a pair $(\hat{C}, \hat{T}) \in G_1^2$ such that $\hat{T} = \hat{C}^z$ (as well as the elements $\hat{K}, \hat{S}, \hat{S}_0, \ldots, \hat{S}_n$ that we added to the public key earlier). For that reason, we will henceforth assume that such a pair is included with the public key σ of the credential scheme. Note that one can view these elements $\hat{K}, \hat{S}, \hat{S}_0, \ldots, \hat{S}_n, \hat{C}, \hat{T}$ as an extra credential of which the numbers $(\kappa, k_0, \ldots, k_n)$ are not known. Therefore the credential scheme remains unforgeable (in the sense that Theorem 14 still holds).

Theorem 15. *The* ShowCredential *protocol is a black-box zero-knowledge proof of knowledge with respect to the language L.*

Proof. The ShowCredential protocol is complete (Theorem 12), and extractable (Theorem 13), so by Definition 22 it remains to show here that there exists a simulator whose behavior is indistinguishable from an honest prover. This simulator $\mathcal{S}$ is given the issuer's public key, a disclosure set $\mathcal{D}$, and a list of attributes k_i for $i \in \mathcal{D}$. We have it proceed as follows.

- It chooses random $\alpha, \beta \in_R \mathbb{Z}_p^*$;
- It sets $\bar{K} = \hat{K}^\alpha$, $\bar{S} = \hat{S}^\alpha$, $\bar{S}_i = \hat{S}_i^\alpha$ for $i = 0, \ldots, n$, and $\tilde{C} = \hat{C}^\beta, \tilde{T} = \hat{T}^\beta$;
- It sends these values to the verifier, and then uses the simulator from the proof of knowledge of the numbers $\beta, \kappa, k_0, (k_i)_{i \in \mathcal{C}}$.

It remains to show that this behavior is indistinguishable from that of honest provers, to any verifier that is given any auxiliary information. First notice that for honest provers and the simulator alike, the elements $\bar{K}$ and $\tilde{C}$ are always randomly distributed in G_1. Also, again for both honest provers and the simulator, the elements $\bar{S}, \bar{S}_i$ and $\tilde{T}$ are determined by $\bar{K}$ and $\tilde{C}$ respectively.

Notice that for any $\beta \in \mathbb{Z}_p^*$ and any set $\mathcal{C}$ of undisclosed attributes, there exist numbers $\kappa, k_0, (k_i)_{i \in \mathcal{C}}$ such that equation (3) holds. Thus the only difference between honest users and the simulator is that an honest user knows these numbers and uses them to honestly prove knowledge of them, while the simulator simulates this proof. However, by the black-box zero-knowledge properties of the proof of knowledge over these numbers, this cannot be detected by the verifier. Thus the verifier can behave no different than it would have done if it had interacted with an honest prover $\mathcal{P}$. □

Theorem 16. *Let* (KeyGen, Issue, ShowCredential) *be an attribute-based credential scheme whose* ShowCredential *protocol is black-box zero-knowledge. Then the scheme is unlinkable.*

Proof. Let the auxiliary input a to the verifier be whatever it learns in the Queries phase of the unlinkability game. In the Challenge phase, instead of performing the showing protocol normally using credential j_b, the challenger uses the simulator $\mathcal{S}$ whose existence is guaranteed by the black-box zero-knowledge property of the ShowCredential protocol. It is clear that in this case the adversary cannot have a non-negligible advantage. By equation (8), then, it also cannot have a non-negligible advantage if the challenger uses credential j_b normally (i.e., without the help of the simulator $\mathcal{S}$). □

Theorem 17. *Our credential scheme is unlinkable.*

Proof. Follows from Theorems 15 and 16. □

5 Performance

5.1 Exponentiation count

Although exponentiations (or scalar multiples, if we had written our groups additively) in elliptic curves are cheap compared to exponentiations in RSA groups, they are still the most expensive action that the prover has to perform. In this section we will therefore count the number of exponentiations the prover has to perform. Recalling that n denotes the total number of attributes, we have

$$n = |\mathcal{C}| + |\mathcal{D}| + 1$$

where $\mathcal{C}, \mathcal{D} \subset \{1, \dots, n\}$ are the indices of the undisclosed and disclosed attributes, respectively.

First note that

$$D = \bar{K}^{-1} \prod_{i \in \mathcal{D}} \bar{S}_i^{-k_i} = C^{-\alpha} \bar{S}^{\kappa} \bar{S}_0^{k_0} \prod_{i \in \mathcal{C}} \bar{S}_i^{k_i}.$$

These expressions for D contain $|\mathcal{D}|$ and $|\mathcal{C}| + 3$ exponentiations, respectively, so if $|\mathcal{C}| + 3 < |\mathcal{D}|$, the prover should use the right hand side to determine D. Moreover, if the prover stores the elements $R := S^{\kappa}$ and $R_i := S_i^{k_i}$ for $i = 0, \dots, n$, then it can calculate D by

$$D = \big(K \prod_{i \in \mathcal{D}} R_i\big)^{-\alpha} = \big(C^{-1} R R_0 \prod_{i \in \mathcal{C}} R_i\big)^{\alpha}$$

both of which take just one exponentiation.

Denote with $\mathrm{pk}(i)$ the amount of exponentiations the prover has to compute in the zero-knowledge proof of knowledge when it presents a DL-representation of length i (in the case of the protocol from Appendix B, we have $\mathrm{pk}(i) = i+6$). Then the number of exponentiations in G_1 that the prover has to do is

- $n+|\mathcal{D}|+\mathrm{pk}(|\mathcal{C}|+3)+5$ exponentiations if $|\mathcal{D}| \le |\mathcal{C}|+3$,
- $n+|\mathcal{C}|+\mathrm{pk}(|\mathcal{C}|+3)+8$ exponentiations if $|\mathcal{D}| \ge |\mathcal{C}|+3$,
- $n + \mathrm{pk}(|\mathcal{C}| + 3) + 6$ if the prover stores and uses R and R_i for $i = 0, \dots, n$.

The prover performs no exponentiations in G_2 and G_T and computes no pairings. This is optimal, since elements from G_2 are bigger and more expensive to deal with than those from G_1 (because generally $G_1 \subset E(\mathbb{F}_q)$ while $G_2 \subset E(\mathbb{F}_{q^k})[p]$, where k is the embedding degree of the curve).

5.2 The Fiat-Shamir heuristic

The zero-knowledge proof from Figure 4 in Appendix B that is used for proving knowledge of the numbers $\beta, \kappa, k_0, (k_i)_{i \in \mathcal{C}}$ consists of four moves. Although our scheme is thus far defined and proven secure in the standard model, if one is willing to assume the random oracle model, then we can apply the Fiat-Shamir heuristic [6, 7, 23] to the Schnorr Σ-protocol for DL-representations (see Figure 3) as follows: the prover receives a nonce $\eta \in_R \mathbb{Z}_p^*$ from the verifier, and uses the protocol from Figure 3 with $c = H(W, D, \eta)$ (for a suitable hash function H). It is easy to see that in the random oracle model, this 2-move protocol is a zero-knowledge proof of knowledge. This essentially reduces the amount of moves in the ShowCredential protocol to just two, and lowers the amount of exponentiations for the prover with 6 (since $\mathrm{pk}(i) = i$ for this proof.)

Concerns have been raised about the security of the Random Oracle Model, however. For example, there exist protocols that are secure in the random oracle model, but do not have any secure standard model instantiation no matter which hash-function is used [16, 26].

5.3 Implementation

We have written a preliminary implementation of our credential scheme, where the prover's performance (in the ShowCredential protocol) on a 320 bit BN curve [5] seems to be on par with the 1024-bit Idemix implementation from the IRMA project (see `irmacard.org` and `github.com/credentials`). Note that according to [32], a 1024-bit modulus corresponds to a 144 bit curve; see also www.keylength.com. Both schemes are implemented in Java running on a 2,7 GHz dual core system, with a total of 6 attributes (including the secret key), and both schemes use the Fiat-Shamir heuristic, resulting in ShowCredential protocols of two moves.

6 Related work

The Idemix credential scheme [12, 31] by Camenisch and Lysyanskaya is probably the most well-known unlinkable attribute-based credential scheme, relying on the difficulty of the strong RSA problem in the group of integers modulo an RSA modulus $n = pq$, of recommended size at least 2048 bits. Although this credential scheme has a lot of desirable properties (it is provably unlinkable and unforgeable), the large size of the modu-

Table 1. Performance comparison for the prover of the ShowCredential protocol of several attribute-based credential schemes. The columns G_{EC}, G_T and G_{RSA} show the amount of exponentiations in elliptic curves, the target group of a bilinear pairing, and RSA groups respectively, while the column labeled e counts the amount of pairings the prover has to compute. Note that exponentiations in G_{RSA} are much more expensive than exponentiations in G_{EC} (assuming their size is such that they offer similar security).

	G_{EC}	G_T	e	G_{RSA}	unlinkable
Our scheme	$n + \|\mathcal{C}\| + 15$	0	0	0	yes
[13]	$2n + 3$	$\|\mathcal{C}\| + 8$	$n + 3$	0	yes
Idemix	0	0	0	$\|\mathcal{C}\| + 3$	yes
U-Prove	$\|\mathcal{C}\| + 1$	0	0	0	no

lus means that, when implementing the prover on smart cards, it is difficult to get acceptable running times for the protocols. For example, in [44] the Idemix showing protocol has been implemented with 4 attributes and n around 1024 bits (while n should really be at least 2048 bits); there the running time for the ShowCredential protocol ranged from 1 to 1.3 seconds, depending on the amount of disclosed attributes.

Another well-known credential scheme is U-Prove [11, 39] by Brands. Based on the difficulty of the discrete logarithm problem in a cyclic group, it can be implemented using elliptic curves, and additionally the showing protocol is much less complicated than that of Idemix, also resulting in more efficiency. However, in U-Prove two transactions executed with the same credential are always linkable, and the showing protocol is only honest-verifier zero-knowledge (i.e., there is no proof that dishonest verifiers cannot extract or learn information about the undisclosed attributes). Moreover, there is no unforgeability proof for U-Prove credentials, and it even seems that no such proof exists under standard intractability assumptions [4]. Table 1 compares the amount of exponentiations of the U-Prove ShowCredential protocol with that of our scheme.

We also mention the "Anonymous Credentials Light" construction from [3], which can also be implemented on elliptic curves, but the credentials are not unlinkable; and [29], which runs in RSA groups like Idemix.

The credential scheme from [13], also by Camenisch and Lysyanskaya, is much closer to the scheme presented here: it is unlinkable, uses elliptic curves and (Type 1) pairings, and uses the LRSW assumption. Camenisch and Lysyanskaya did not include a showing protocol that allows attributes to be disclosed (that is, it is assumed that all attributes are kept secret), but it is not very difficult to keep track of how much less the prover has to do if he voluntarily discloses some attributes. In Table 1 we compare the amount of exponentiations and pairings that the prover has to perform in the ShowCredential protocol of this scheme to ours (assuming that the protocol from Appendix B is used for the proofs of knowledge of DL-representations). We see that the amount of exponentiations the prover has to perform in the ShowCredential protocol of [13] is roughly 1.5 times as large as in our scheme. Since, additionally, computing pairings is significantly more expensive than exponentiating, we expect our credential scheme to be at least twice as efficient.

Finally, a blindable version of U-Prove was recently proposed in [30]. Although an unlinkable credential scheme is aimed at, the paper contains no unlinkability proof. Moreover, it turns out that the scheme is forgeable: if sufficiently many provers collide then they can create new credentials containing any set of attributes of their choice, without any involvement of the issuer [42].

Table 1 compares the amount of exponentiations in our scheme to those of [13], U-Prove and Idemix. However, note that the comparison between Idemix and the other schemes is not very strong since RSA exponentiations are significantly more expensive than exponentiations in elliptic curves. Also, the U-Prove showing protocol offers no unlinkability.

7 Conclusion

In this paper we have defined a new self-blindable attribute-based credential scheme, and given a full security proof by showing that it is unforgeable and unlinkable. Our scheme is based on a standard hardness assumption and does not need the random oracle model. Based on the fact that it uses elliptic curves and bilinear pairings (but the latter only on the verifier's side) and on a comparison of exponentiation counts, we have shown it to be more efficient than comparable schemes such as Idemix and the scheme from [13], achieving the same security goals at less cost. In the future, we plan to implement the scheme on a smart card to measure the running time of the ShowCredential protocol, for an explicit comparison with other schemes (c.f. [37, 44]).

References

[1] A. Antipa, D. Brown, A. Menezes, R. Struik, and S. Vanstone. Validation of elliptic curve public keys. In Y. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *LNCS*, pages 211–223. Springer Berlin Heidelberg, 2002. 10.1007/3-540-36288-6_16.

[2] G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable rfid tags via insubvertible encryption. In *Proceedings of the 12th ACM Conference on Computer and Communications Security - CCS '05*, pages 92–101, New York, NY, USA, 2005. ACM. 10.1145/1102120.1102134. URL http://doi.acm.org/10.1145/1102120.1102134.

[3] F. Baldimtsi and A. Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1087–1098, New York, NY, USA, 2013. ACM. 10.1145/2508859.2516687.

[4] F. Baldimtsi and A. Lysyanskaya. On the security of one-witness blind signature schemes. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *LNCS*, pages 82–99. Springer Berlin Heidelberg, 2013. 10.1007/978-3-642-42045-0_5.

[5] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *LNCS*, pages 319–331. Springer Berlin Heidelberg, 2006. 10.1007/11693383_22.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993. 10.1145/168588.168596.

[7] D. Bernhard, O. Pereira, and B. Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In X. Wang and K. Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 626–643. Springer Berlin Heidelberg, 2012. 10.1007/978-3-642-34961-4_38.

[8] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference - ITCS '12*, pages 326–349, New York, NY, USA, 2012. ACM. 10.1145/2090236.2090263.

[9] I. F. Blake, G. Seroussi, and N. P. Smart, editors. *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005. Cambridge Books Online.

[10] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008. 10.1007/s00145-007-9005-7.

[11] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000.

[12] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer Berlin Heidelberg, 2001. 10.1007/3-540-44987-6_7.

[13] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer Berlin Heidelberg, 2004. 10.1007/978-3-540-28628-8_4.

[14] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. S. J. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *LNCS*, pages 410–424. Springer Berlin Heidelberg, 1997. 10.1007/BFb0052252.

[15] J. Camenisch, S. Hohenberger, and M. Ø. Pedersen. Batch verification of short signatures. In M. Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, pages 246–263, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 10.1007/978-3-540-72540-4_14. URL https://eprint.iacr.org/2007/172.pdf.

[16] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. 10.1145/1008731.1008734.

[17] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology - CRYPTO '92*, volume 740 of *LNCS*, pages 89–105. Springer, 1993.

[18] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO '94*, pages 174–187, London, UK, UK, 1994. Springer-Verlag. 10.1007/3-540-48658-5_19.

[19] R. Cramer, I. Damgård, and P. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *LNCS*, pages 354–372. Springer Berlin Heidelberg, 2000. 10.1007/978-3-540-46588-1_24.

[20] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *LNCS*, pages 445–456. Springer, 1991. 10.1007/3-540-46766-1_36.

[21] I. Damgård. On σ-protocols, 2010. URL http://www.cs.au.dk/~ivan/Sigma.pdf. CPT 2010, v.2.

[22] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing - STOC '90*, pages 416–426. ACM, 1990. 10.1145/100216.100272.

[23] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO' 86*, volume 263 of *LNCS*, pages 186–194. Springer Berlin Heidelberg, 1987. 10.1007/3-540-47721-7_12.

[24] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16): 3113–3121, 2008. 10.1016/j.dam.2007.12.010.

[25] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[26] S. Goldwasser and Y. T. Kalai. On the (in)security of the fiat-shamir paradigm. In *44th Symposium on Foundations of Computer Science - FOCS 2003*, pages 102–113. IEEE Computer Society, 2003. 10.1109/SFCS.2003.1238185.

[27] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, apr 1988. 10.1137/0217017.

[28] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 10.1137/0218012.

[29] J. Hajny and L. Malina. Unlinkable attribute-based credentials with practical revocation on smart-cards. In S. Mangard, editor, *Smart Card Research and Advanced Applications*, volume 7771 of *LNCS*, pages 62–76. Springer Berlin Heidelberg, 2013. 10.1007/978-3-642-37288-9_5.

[30] L. Hanzlik and K. Kluczniak. A short paper on how to improve U-Prove using self-blindable certificates. In N. Christin and R. Safavi-Naini, editors, *Financial Cryptography and Data Security: 18th International Conference, FC 2014*, pages 273–282, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. 10.1007/978-3-662-45472-5_17.

[31] IBM Research Zürich Security Team. Specification of the Identity Mixer cryptographic library, version 2.3.0. Technical report, IBM Research, Zürich, feb 2012. URL https://tinyurl.com/idemix-spec.

[32] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001. 10.1007/s00145-001-0009-4.

[33] Y. Lindell. Sigma protocols and zero-knowledge. Lecture notes from Winter School on Secure Computation and Efficiency, 2011. URL http://u.cs.biu.ac.il/~lindell/winterschool2011/lecture%205.pdf.

[34] A. Lysyanskaya. Pseudonym systems. Master's thesis, Massachusetts Institude of Technology, 1999. URL https://groups.csail.mit.edu/cis/theses/anna-sm.pdf.

[35] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99*, pages 184–199, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 10.1007/3-540-46513-8_14.

[36] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(5):1234–1243, 2001.

[37] W. Mostowski and P. Vullers. Efficient u-prove implementation for anonymous credentials on smart cards. In M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, editors, *Security and Privacy in Communication Networks*, volume 96 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 243–260. Springer Berlin Heidelberg, 2012. 10.1007/978-3-642-31909-9_14.

[38] M. Naor. On cryptographic assumptions and challenges. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 96–109, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 10.1007/978-3-540-45146-4_6.

[39] C. Paquin and G. Zaverucha. U-prove cryptographic specification v1.1 (revision 3). Released under the Open Specification Promise, December 2013. URL http://research.microsoft.com/apps/pubs/default.aspx?id=166969.

[40] C. Schnorr. Efficient identification and signatures for smart cards. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, volume 434 of *LNCS*, pages 688–689. Springer Berlin Heidelberg, 1990. 10.1007/3-540-46885-4_68.

[41] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 256–266. Springer Berlin Heidelberg, 1997. 10.1007/3-540-69053-0_18.

[42] E. Verheul, S. Ringers, and J.-H. Hoepman. The self-blindable U-Prove scheme from FC'14 is forgeable. Cryptology ePrint Archive, Report 2015/725, 2015. https://eprint.iacr.org/2015/725.

[43] E. R. Verheul. Self-blindable credential certificates from the weil pairing. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT*, volume 2248 of *LNCS*, pages 533–551. Springer, 2001.

[44] P. Vullers and G. Alpár. Efficient selective disclosure on smart cards using idemix. In S. Fischer-Hübner, E. de Leeuw, and C. Mitchell, editors, *Policies and Research in Identity Management*, volume 396 of *IFIP Advances in Information and Communication Technology*, pages 53–67. Springer Berlin Heidelberg, 2013. 10.1007/978-3-642-37282-7_5.

[45] C. Wachsmann, L. Chen, K. Dietrich, H. Löhr, A.-R. Sadeghi, and J. Winter. Lightweight anonymous authentication with tls and daa for embedded mobile devices. In M. Burmester, G. Tsudik, S. Magliveras, and I. Ilić, editors, *Information Security: 13th International Conference, ISC 2010*, pages 84–98, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 10.1007/978-3-642-18178-8_8. URL https://eprint.iacr.org/2011/101.pdf.

[46] V. K. Wei and T. H. Yuen. More short signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005:463, 2005. URL http://eprint.iacr.org/2005/463.

A Definitions

Definition 18 (DL assumption). Let G be a cyclic group of order p, and let $\ell = |p|$. The Discrete Logarithm (DL) assumption holds in G if no probabilistic polynomial-time algorithm can, given a generator X and an element $K = X^k$, output k with probability that is non-negligible in ℓ.

Definition 19 (DL-representations). Let G be a cyclic group of prime order p and let $K_1, \ldots, K_m \in G$ all be distinct. If for some $K \in G$ the numbers $(k_1, \ldots, k_m) \in \mathbb{Z}_p^m$ are such that $K = \prod_{j=1}^m K_j^{k_j}$, then $(k_1, \ldots, k_m)$ is called a *DL-representation* of K with respect to $(K_1, \ldots, K_m)$. When $k_1 = \cdots = k_m = 0$ we say that the DL-representation is trivial.

The following shows that the problem of creating such DL-representations is equivalent with the discrete-logarithm problem. A proof of this statement may be found in, for example, [11, p. 60].

Proposition 20. *Let $(K_1, \ldots, K_m)$ be randomly distributed. No probabilistic polynomial-time algorithm can, on input $(K_1, \ldots, K_m)$, generate a non-trivial DL-representation of $1 \in G$ with respect to $(K_1, \ldots, K_m)$ with non-negligible probability, assuming that the discrete logarithm-problem holds in G.*

This proposition implies that if a polynomial-time algorithm finds such a DL-representation of 1, then the representation is trivial. The following extension of this result to bilinear groups can be proven using an almost identical proof.

Proposition 21. *Let $e\colon G_1 \times G_2 \to G_T$ be a Type 3 pairing in which the following problem is intractable: given $P, P^a \in G_1, Q, Q^a \in G_2$ for generators P, Q, compute a. Let the tuple $X_1, \ldots, X_m \in G_1$ and the tuple $Y_1, \ldots, Y_m$ be such that $\log_P X_j = \log_Q Y_j$, i.e., $e(P, Y_j) = e(X_j, Q)$. Then no probabilistic polynomial-time algorithm can, on input $X_1, \ldots, X_m, Y_1, \ldots, Y_m$ generate a non-trivial DL-representation of $1 \in G_1$ with respect to $(X_1, \ldots, X_m)$.*

Note that the problem mentioned above (given P, P^a, Q, Q^a, compute a) is implied by the LRSW and whLRSW assumptions (as well as, for example, the q-SDH and the SDH assumptions [10], various variants of the co-CDH assumptions, and a number of other pairing-specific variants of the Diffie-Hellman problem). The only difference with Proposition 20 is that the adversary now also has $Y_1, \ldots, Y_m$ to work with.

Let $\mathcal{A}$ and $\mathcal{B}$ be two interactive algorithms. Then we denote with $\mathrm{view}_{\mathcal{A}}(\mathcal{A}(x, a) \leftrightarrow \mathcal{B}(x, w))$ a random variable containing x, a, the random tape of $\mathcal{A}$, and the messages that $\mathcal{A}$ receives during a joint conversation with $\mathcal{B}(x, w)$.

Definition 22 (Black-box zero-knowledge proof of knowledge). Let L be a language and let R be a polynomially computable relation for L. Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system for L. We say that $(\mathcal{P}, \mathcal{V})$ is a *black-box zero-knowledge proof of knowledge* for R ([28], see also [19, 25]) if there exists an expected polynomial-time simulator $\mathcal{S}$ and a polynomial-time extractor χ, that satisfy the following conditions:

Completeness For all x, w such that $R(x, w) = 1$,

$$\Pr[\mathcal{P}(x, w) \leftrightarrow \mathcal{V}(x) \to 1] = 1.$$

Black-box zero-knowledge For any probabilistic polynomial-time Turing machine $\mathcal{V}^*$, and for any auxiliary input $a \in \{0,1\}^*$ of polynomial length in $|x|$, we have

$$\begin{aligned}&\{\mathrm{view}_{\mathcal{V}^*}(\mathcal{P}(x, w) \leftrightarrow \mathcal{V}^*(x, a))\}_{x \in L,\, a \in \{0,1\}^*} \\ &\quad \overset{c}{\approx} \{\mathrm{view}_{\mathcal{V}^*}(\mathcal{S}(x) \overset{\blacksquare}{\rightarrow} \mathcal{V}^*(x, a))\}_{x \in L,\, a \in \{0,1\}^*}\end{aligned} \tag{7}$$

for any w such that $(x, w) \in R$ (the symbol $\overset{c}{\approx}$ denotes computational indistinguishability.)

Black-box extraction Let $x \in L$. For any probabilistic polynomial-time Turing machine $\mathcal{P}^*$, and for any auxiliary input $a \in \{0,1\}^*$ of polynomial length in $|x|$, if

$$\Pr[\mathcal{P}^*(x, a) \leftrightarrow \mathcal{V}(x) \to b : b = 1] \geq \epsilon(|x|)$$

for some function $\epsilon\colon \mathbb{N} \to [0, 1]$, then there exists a negligible function ν such that

$$\begin{aligned}&\Pr[\mathcal{P}^*(x, a) \overset{\blacksquare}{\leftarrow} \chi(x) \to w : R(x, w) = 1] \\ &\qquad\qquad \geq \epsilon(|x|) - \nu(|x|).\end{aligned}$$

In other words, if the prover convinces the verifier often that it knows some witness for $x \in L$, then the extractor computes a witness for x almost as often.

We will sometimes use that the second property, black-box zero-knowledge, implies (and is in fact equivalent with) the following: for any $x \in L$, any w such that $(x, w) \in R$, and any auxiliary input a:

$$\Big| \Pr[\mathcal{P}(x, w) \leftrightarrow \mathcal{V}^*(x, a) \to 1] - \Pr[\mathcal{S}(x) \overset{\blacksquare}{\rightarrow} \mathcal{V}^*(x, a) \to 1] \Big| < \mathrm{negl}(|x|). \tag{8}$$

That is, if using a witness w the prover can get verifier $\mathcal{V}^*$ to accept, then the simulator $\mathcal{S}$ can, given black-box access to the verifier, make him accept without knowing the witness w. Indeed, suppose some machine $\mathcal{V}^*$ violates the formula above, i.e., there exist x, w, a such that with non-negligible probability $\mathcal{V}^*(x, a)$ outputs 0 when it interacts with $\mathcal{S}(x)$ and 1 when interacting with $P(x, w)$. Note that the output of $\mathcal{V}^*$ can be calculated in polynomial time from its view. Consider then the distinguisher that, given such a view, returns the output of $\mathcal{V}^*$ corresponding to this view. This distinguisher would then, for these particular x, w, a, be able to distinguish $\text{view}_{\mathcal{V}^*}(\mathcal{P}(x,w) \leftrightarrow \mathcal{V}^*(x,a))$ from $\text{view}_{\mathcal{V}^*}(\mathcal{S}(x) \overset{\blacksquare}{\rightarrow} \mathcal{V}^*(x,a))$, violating equation (7).

Definition 23. We define existential unforgeability of a signature scheme (KeyGen, $\mathsf{Sign}, \mathsf{Verify}$) under adaptive chosen message attacks in terms of the following game [27]. It is a game between an adversarial user $\mathcal{A}$ and a signer $\mathcal{S}$, controlled by the challenger. The game proceeds as follows.

Setup The challenger generates a private-public key pair $(SK, PK) = \mathsf{KeyGen}(1^k)$. It sends PK to the adversary $\mathcal{A}$.

Queries The adversary requests signatures on messages $m_1, \dots, m_q$ that it may choose adaptively. The challenger responds to each query with a signature $\sigma_i \leftarrow \mathsf{Sign}(SK, m_i)$.

Output The adversary $\mathcal{A}$ outputs a pair (m, σ) and wins the game if σ is a valid signature over m, and $m \neq m_i$ for all $1 \leq i \leq q$.

When no probabilistic polynomial-time algorithm can win this game with non-negligible probability we say that the signature scheme is existentially unforgeable under adaptive chosen-message attacks.

B Proofs of knowledge of discrete logarithms and DL-representations

In the protocols presented in this paper, proofs of knowledge of DL-representations occur. The Schnorr Σ-protocol [40] and its extension to DL-representations (see Figure 3 and [11]) will not suffice here for those, because they are only honest-verifier zero-knowledge, and we need our proof to be zero-knowledge even against verifiers that are not honest (as in Definition 22).

Fortunately, there are extensions of the Schnorr Σ-protocols that are fully zero-knowledge; we present an optimization of the one from [19] for DL-representations here. When $n = 1$ is taken, it reduces to a proof of knowledge for discrete logarithms.[8]

8 Other protocols satisfying Definition 22 that could be used are, for example, the one sketched in [21, p. 16] which also has 4 moves, or the protocol for discrete logs from [33, p. 31], which is easily extended to a protocol for DL-representation but has 5 moves.

Let G be a cyclic group of prime order p in which the DL-problem is intractable. Recall that a trace (W, c, z) of the Schnorr Σ-protocol for K is valid if $P^z = WK^c$. Such a trace can be generated by choosing z, w randomly, setting $W = P^z K^{-c}$, and returning (W, c, z). The output of this algorithm is distributed identically to actual traces for $K = P^k$.

Suppose the prover $\mathcal{P}$ wants to prove knowledge of the DL-representation $(k_1, \dots, k_n)$ of $K = P_1^{k_1} \cdots P_n^{k_n}$ with respect to $P_1, \dots, P_n \in G$, where all P_i are distinct and unequal to 1. The proof consists of two parts, each having three moves:

- $\mathcal{V}$ uses the simulator above for K with respect to P_1, obtaining (W, c, z). It sends W to $\mathcal{P}$, and executes the Schnorr 3-move Σ-protocol on the numbers c, z such that $W = P_1^z K^{-c}$.
- $\mathcal{P}$ uses the OR-protocol from [18] (a Σ-protocol consisting of 3 moves) to show that it knows either numbers c', z' such that (W, c', z') is accepting for K (with P_1 as base point), or the DL-representation $(k_1, \dots, k_n)$ of K, without revealing which.

The second and third moves of the first part can then be executed simultaneously with the first and second moves of the second part, resulting in a protocol of 4 moves with $n + 6$ exponentiations for the prover. The protocol is shown in Figure 4; for readability the protocol has not been contracted there to 4 moves. We emphasize that in implementations, moves 2 and 4 and moves 3 and 5 should be done simultaneously.

We briefly sketch the existence of a simulator and extractor for this protocol. First we remark that proofs from the first part and second part are both witness-hiding [22].

The simulator uses its black-box access to the verifier to extract the numbers c, z from the first part. Then it can use these in the OR-protocol in the second part to prove that it either knows (c, z) or $(k_1, \dots, k_n)$. Since the verifier cannot detect that it is being rewinded and since the OR-protocol is witness-hiding, the verifier cannot tell which of the two statements the simulator is proving. Therefore the simulator's behavior is indistinguishable from honest provers.

The extractor runs the first part of the protocol normally. Then, using its black-box access to $\mathcal{P}$ it can rewind $\mathcal{P}$ in the second part, and obtain either $(k_1, \dots, k_n)$ of K (in which case we are done), or numbers c', z' such that (W, c', z') is accepting for

Common information: $P_1, \ldots, P_n, K \in G$; the group G		
Prover		**Verifier**
knows k_i such that $K = \prod_{i=1}^{n} P_i^{k_i}$		
random $w_1, \ldots, w_n \in_R \mathbb{Z}_p^*$		
send $W = P_1^{w_1} \cdots P_n^{w_n}$	$\longrightarrow$	
	$\longleftarrow$	send $c \in_R \mathbb{Z}_p$
$\forall i \in [1, n]$ send $s_i = ck_i + w_i$	$\longrightarrow$	
		verify $K^c W \overset{?}{=} \prod_{i=1}^{n} P_i^{s_i}$

Fig. 3. The Schnorr Σ-protocol for DL-representations.

Common information: $P_1, \ldots, P_n, K \in G$; the group G		
Prover		**Verifier**
knows k_i such that $K = \prod_{i=1}^{n} P_i^{k_i}$		
	Phase 1	
$c_1, e, z'_1, z'_2, \bar{w}_1, \ldots, \bar{w}_n \in_R \mathbb{Z}_p^*$		$c, z, w_1, w_2 \in_R \mathbb{Z}_p^*$
	$\longleftarrow$	send $W = P_1^z K^{-c}, \tilde{W} = P_1^{w_1} K^{w_2}$
send e	$\longrightarrow$	
	$\longleftarrow$	send $z_1 = w_1 + ez$, $z_2 = w_2 - ec$
verify $P_1^{z_1} K^{z_2} \overset{?}{=} \tilde{W} W^e$		
	Phase 2	
send $W' = P_1^{z'_1} K^{z'_2} W^{-c_1}$, $\bar{W} = P_1^{\bar{w}_1} \cdots P_n^{\bar{w}_n}$	$\longrightarrow$	
	$\longleftarrow$	send c
set $c_2 = c - c_1$		
$\forall i$ set $\bar{z}_i = \bar{w}_i + c_2 k_i$		
send $c_1, c_2, z'_1, z'_2, (\bar{z}_i)_{i=1,\ldots,n}$	$\longrightarrow$	
		verify $c \overset{?}{=} c_1 + c_2$
		verify $P_1^{z'_1} K^{z'_2} \overset{?}{=} W' W^e$
		verify $P_1^{\bar{z}_1} \cdots P_n^{\bar{z}_n} \overset{?}{=} \bar{W} K^{c_2}$

Fig. 4. Black-box zero-knowledge proof of knowledge of a DL-representation.

K (with P_1 as base point). Suppose the latter is the case. Since the Σ-protocol executed in the first part is witness-hiding, the new trace (W, c', z') will with overwhelming probability differ from the the trace (W, c, z) that the extractor generated itself. Thus it sets $k' = (z - z')/(c - c')$, so that $K = P_1^{k'}$. Now the extractor outputs $(k', 0, \ldots, 0)$ (which is also a DL-representation of K with respect to $P_1, \ldots, P_n$).

C Multi-show and issuer unlinkability

In the unlinkability game from Definition 3, the adversary plays the role of the issuer in the Setup phase and the role of the verifier in the Challenge phase. This definition of unlinkability implies both multi-show unlinkability and issuer unlinkability, as follows.

Multi-show unlinkability Suppose there exists a malicious verifier that can link two transactions as in the Challenge phase of our unlinkability game, without itself having issued the credentials from those transactions. Then there certainly also exists an adversary that, by using this verifier, breaks blindness in the sense of Definition 3. Thus unlinkability as in Definition 3 implies multi-show unlinkability.

Issuer unlinkability Consider the following game for issuer unlinkability. The Setup and Queries phases are as in Definition 3, but the Challenge and Output phases are as follows:

Challenge The adversary chooses a credential j and a disclosure set $\mathcal{D} \subset \{1, \ldots, n\}$, and informs the challenger of its choice. The challenger flips a bit $b \in_R \{0, 1\}$, takes $j_0 \in_R \{1, \ldots, m\}$, and sets $j_1 = j$. Next it engages in the ShowCredential protocol with the adversary on credential j_b, acting as the prover. All attributes whose index is in j are disclosed.

Output The adversary outputs a bit b' and wins if $b = b'$.

If there exists an adversary that can win this game, then there also exists an algorithm that breaks blindness in the sense of Definition 3: if an algorithm can win this game with non-negligible probability, it means that it can distinguish credential $j = j_1$ from any other credential $j \in_R \{1, \dots, m\}$, so that it could certainly also distinguish j_1 from a fixed j_0.

Essentially, the reason why these variations of the unlinkability game imply unlinkability in the sense of Definition 3, is because in both of them the adversary is endowed with less power. Indeed, in the first case it does not know the issuer's secret key, and since it did not issue the credentials it does not have access to the issuer's view of the executions of the Issue protocol; and in the second case it does not get to choose the credential j_1.