An efficient self-blindable attribute-based credential scheme

Sietse Ringers, Eric Verheul, and Jaap-Henk Hoepman

Radboud University, Nijmegen, The Netherlands
{sringers,e.verheul,jhh}@cs.ru.nl

Abstract. An attribute-based credential scheme allows a user, given a set of attributes, to prove ownership of these attributes to a verifier, voluntarily disclosing some of them while keeping the others secret. A number of such schemes exist, of which some additionally provide unlinkability: that is, when 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 implementations 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 on the user's side.

Keywords: attribute-based credentials, unlinkable, self-blindable, elliptic curves, bilinear pairings

1 Introduction

An attribute-based credential (ABC) scheme allows a user, given a set of attributes $k_1, \ldots, k_n$, 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, when reusing a credential 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 ABC schemes are Idemix [14,29] and U-Prove [12,35]. However, to date there is no provably secure scheme that is sufficiently efficient to allow truly secure 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[1] 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* [39]: 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 in which the verifier learns nothing at all about the credential besides the attributes that are disclosed (and the fact that the credential is valid). In fact, the showing protocol is a zero-knowledge proof of knowledge. The scheme does not rely on the random oracle model (although usage of this model can lead to a performance increase through the Fiat-Shamir heuristic [21]), 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. However, the user only needs to perform computations on the cheaper of the two. These two facts ensure that the amount of work that the user has to perform is minimal, and indeed our scheme is cheaper for the user than any comparable scheme that we know of.

The unforgeability of our credential scheme will be implied by the LRSW assumption [15,32,31] introduced by Lysyanskaya et al., and used in many subsequent works (for example, [15,42,41,13,2]). Actually, for our purposes a weaker (in particular, non-interactive and thus falsifiable [34]) version of this assumption called the whLRSW assumption [42] 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 Section 5 we will discuss the performance of our scheme, by counting the amount of exponentiations that the user has to perform and by showing average runtimes of an implementation of our scheme. First, we briefly review and compare a number of other attribute-based credential schemes, in terms of features, efficiency and speed, and security.

1.1 Related work

The Idemix credential scheme [14,29] by Camenisch and Lysyanskaya is probably the most well-known unlinkable attribute-based credential scheme, relying

[1] `https://www.irmacard.org`

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 modulus means that, when implementing the user on smart cards, it is difficult to get acceptable running times for the protocols. For example, in [40] 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 [12,35] 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].

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 [27], which runs in RSA groups like Idemix.

The credential scheme from [15], 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. However, when showing a credential the user has to compute an amount of pairings that is linear in the amount of disclosed attributes.

Finally, a blindable version of U-Prove was recently proposed in [28]. 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 users collide then they can create new credentials containing any set of attributes of their choice, without any involvement of the issuer [38].

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 tape (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 $\mathcal{A}\overset{\blacksquare}{\rightarrow}\mathcal{B}$ 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 we will use the Camenisch-Stadler notation [16]. 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 $K = P_1^{k_1} P_2^{k_2}$. (Unlike Camenisch and Stadler, we do not use Greek letters for the unknowns; instead we will consistently write them on the right-hand side of the equation.) Such proofs are based on standard techniques and occur in many areas of cryptography. In our case the protocol from [19] could for example be used.

Definition 1. An attribute-based credential scheme consists of the following protocols. (We assume a single issuer, but this can easily be generalized to multiple issuers.)

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

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

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

Here k_0 is the user's private key, that is to be chosen from the attribute space M by the user; the Issue protocol should prevent the issuer from learning it. We assume that before execution of this protocol, the issuer and user have reached agreement on the values of the attributes $k_1, \dots, k_n$. The secret key and attributes $k_0, k_1, \dots, k_n$ are contained in the credential c.[2]

ShowCredential An interactive protocol between a user $\mathcal{P}$ and verifier $\mathcal{V}$ which is such that, if c is a credential[3] issued using the Issue protocol over attributes $(k_1, \dots, k_n)$ using private signing key s corresponding to public key σ, then for any disclosure set $\mathcal{D} \subset \{1, \dots, n\}$ the user can make the verifier accept:

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

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

[2] 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.

[3] 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 4.3), and to proving that the hidden attributes satisfy arbitrary linear combinations [12].

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

- *Unforgeability* (see Definition 2): no user can prove possession of attributes that were not issued to it by the issuer.
- *Multi-show unlinkability* (see Definition 3): If a verifier $\mathcal{V}$ participates in the ShowCredential protocol twice, in which the same credential was involved, 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.
- *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.

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

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 user 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 user, 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 B).

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.

- $\mathsf{\textbf{Issue}}(k_{1,j}, \dots, k_{n,j})$ The adversary chooses a set of attributes $(k_{1,j}, \dots, k_{n,j})$, 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 $(k_{1,j}, \dots, k_{n,j})$.
- $\mathsf{\textbf{ShowCredential}}(j, \mathcal{D})$ The adversary and challenger engage in the showing protocol on credential j, the challenger acting as the user and the adversary as the verifier. Each time the adversary may choose the disclosure set $\mathcal{D}$.
- $\mathsf{\textbf{Corrupt}}(j)$ 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 $\mathcal{D} \subset \{1, \dots, n\}$. 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 user, it engages in the $\mathsf{ShowCredential}$ 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 $\mathsf{ShowCredential}$ protocol, as long as they disclosed the same attributes with the same values. We will achieve this for our scheme by proving that our $\mathsf{ShowCredential}$ protocol is *black-box zero-knowledge* (see Definition 22), which essentially means that the verifier learns nothing at all besides the statement that the user 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 *bi-*

linear 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 [5,33]. Generally elements from G_1 are smaller and more efficient to do computations with in such group pairs. For more information about bilinear group pairs and pairings we refer to [22]; see also, for example, Chapters I and X from [10].

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 users or multiple credentials. For example, the number $k_{i,j}$ will refer to the i-th attribute of the credential of user 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 $\nu(\ell) < \mathrm{negl}(\ell)$ 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* [42], which as we will show below, is implied by the LRSW assumption [31,32] introduced by Lysyanskaya, Rivest, Sahai, and Wolf. The latter assumption has been proven to hold in the generic group model [37], and has been used in a variety of schemes (for example, [15,42,41,13,2]). 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 [34]. 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 $T = K^{z+\kappa a z}$, 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 $(p, e, G_1, G_2, G_T, Q, Q^a, Q^z)$ where $Q \in_R G_2 \setminus \{1\}$, along with 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

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

where L 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 $\{(\kappa_j, K_j, K_j^a, K_j^{z+\kappa_j az})\}_{j=1,\dots,q}$ 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 [42]) 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 $\kappa \notin \{\kappa_1, \dots, \kappa_q\}$. 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

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

where the probability is over the choice of $a, z,\ \kappa_1, \dots, \kappa_q,\ K_1, \dots, K_q, Q$, 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., [11]). 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 [9] 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 could not be 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 Section 4 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 [18] 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.

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 $a, a_0, \ldots, a_n, z \in_R \mathbb{Z}_p^*$ and sets $A = Q^a, A_0 = Q^{a_0}, \ldots, A_n = Q^{a_n}$, and $Z = Q^z$. The public key is the tuple $\sigma = (p, e, Q, A, A_0, \ldots, A_n, Z)$ and the private key is the tuple $(a, a_0, \ldots, a_n, z)$.

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

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

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

The numbers $k_n \in \mathbb{Z}_p$ are the attributes. Although p may vary each time the KeyGen$(1^\ell, n)$ algorithm is invoked on a fixed security parameter ℓ, the attribute space $\mathbb{Z}_p$ will always contain $\{0, \ldots, 2^{\ell-1}\}$. In our credential scheme in section 4, the zeroth attribute k_0 will serve as the user'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 $(k_0, \ldots, k_n)$ and the elements $(K, S, S_0, \ldots, S_n)$), we will often think of it as if it is. Finally, we call a message-signature pair, i.e., a tuple of the form $((k_0, \ldots, k_n), (\kappa, K, S, S_0, \ldots, S_n, T))$ where $(\kappa, K, S, S_0, \ldots, S_n, T)$ is a valid signature over $(k_0, \ldots, k_n)$, a *credential*.

Notice that if $(k_0, \ldots, k_n), (\kappa, K, S, S_0, \ldots, S_n, T)$ is a valid credential, then $(k_0, \ldots, k_n), (\kappa, K^\alpha, S^\alpha, S_0^\alpha, \ldots, S_n^\alpha, T^\alpha)$ for any $\alpha \in \mathbb{Z}_p^*$ is another valid credential having the same attributes. That is, in the terminology of Verheul [39] 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 Theorem 10 essentially says that this is impossible. This self-blindability is what makes this signature scheme suitable for the purpose of creating an unlinkable ShowCredential protocol.

The number κ will play a critical role in the unforgeability proof of our signature and credential schemes (Theorem 14).[4]

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

[4] In the Idemix credential scheme [14], 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.

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

User		**Issuer**
knows secret key k_0		knows $a, a_0, \ldots, a_n$, z
		choose $\bar{K} \in_R G_1$
	$\longleftarrow$	send $\bar{S} = \bar{K}^a, \bar{S}_0 = \bar{K}^{a_0}$
choose $\alpha, \kappa' \in_R \mathbb{Z}_p^*$		
set $S = \bar{S}^\alpha, S_0 = \bar{S}_0^\alpha$		
send $S, S_0, R = S^{\kappa'} S_0^{k_0}$	$\longrightarrow$	
$\mathrm{PK}\{(\kappa', k_0)\colon R = S^{\kappa'} S_0^{k_0}\}$	$\longleftrightarrow$	
		set $K = S^{1/a}$
		verify $S \neq \bar{S}, K = S_0^{1/a_0}$
		choose $\kappa'' \in_R \mathbb{Z}_p$
		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 $\kappa'', K, S_1, \ldots, S_n, T$
set $\kappa = \kappa' + \kappa''$		
$\mathsf{Verify}((k_0, \ldots, k_n), (\kappa, K, S, S_0, \ldots, S_n, T), \sigma)$		
return $(k_0, \ldots, k_n), (\kappa, K, S, S_0, \ldots, S_n, T)$		

Fig. 1. The Issue protocol. In the protocol, the issuers sends two elements $\bar{S}, \bar{S}_0$ (having the appropriate relative discrete log) to the user, who blinds them using a random number, and sends the blinded versions to the issuer. With respect to these blinded elements, the user then proves that it knows its secret key k_0 and its contribution κ' to the number κ. If the verifier is convinced, it chooses its own contribution κ'' to κ, and it computes the remaining elements $K, S_1, \ldots, S_n, T$ such that $(\kappa' + \kappa'', K, S, S_0, \ldots, S_n, T)$ is a valid signature over the attributes. These elements are sent to the user who finally constructs the credential.

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 user, and a showing protocol, which allows a user 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 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 $\mathcal{D} \subset \{1, \ldots, n\}$ for the index set of the disclosed attributes, and

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

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

User		**Verifier**
knows $K, S, S_0, \ldots, S_n, \kappa, (k_i)_{i\in\mathcal{C}}, C, T$		
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 $\bar{K}$, $\bar{S}$, $(\bar{S}_i)_{i=0,\ldots,n}$, $\tilde{C}$, $\tilde{T}$	$\longrightarrow$	
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 $e(\bar{K}, A) \overset{?}{=} e(\bar{S}, Q)$
		and $e(\bar{K}, A_i) \overset{?}{=} e(\bar{S}_i, Q) \ \forall i \in [0, n]$
		and $e(\tilde{C}, Z) \overset{?}{=} e(\tilde{T}, Q)$

Fig. 2. The ShowCredential protocol. We assume that the user has the element $C = KS^\kappa S_0^{k_0} \cdots S_n^{k_n}$ stored so that it does not need to compute it every time the protocol is run (see Section 5 for more such optimizations). In the protocol, he user first blinds K, S and each S_i with a random number, and C and T with a different random number, resulting in new elements $\bar{K}, \bar{S}, \bar{S}_i$ and $\tilde{C}, \tilde{T}$. These are sent to the verifier. Then, the user proves that he knows the hidden attributes and the number κ, as well as a number β which is such that $\tilde{C}^\beta$ is of the required form $\tilde{C}^\beta = \bar{K}\bar{S}^\kappa \bar{S}_0^{k_0} \prod_{i=1}^n \bar{S}_i^{k_i}$. If the proof of knowledge is valid and the elements $\bar{K}$, $\bar{S}$ and $\bar{S}_i$ on the one hand and $\tilde{C}$, $\tilde{T}$ on the other hand have the appropriate relative discrete logarithms (which the verifier checks by calculating a number of pairings), then the verifier accepts.

for the index set of the undisclosed attributes. We do not consider the index 0 of the secret key k_0 to be part of this set, as it is always kept secret.

The Issue protocol is such that both parties contribute to κ and K with neither party being able to choose the outcome in advance (unlike the signing algorithm of the signature scheme from the previous section, where the signer chooses κ and K on its own). This ensures that these elements are randomly distributed even if one of the parties is dishonest. Additionally, the issuer is prevented from learning the values of κ and the secret key k_0.

As noted earlier, we assume that the user and issuer have agreed on the attributes $k_1, \ldots, k_n$ to be contained in the credential before executing this protocol. Similarly, we assume that the user 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 elliptic curve group. 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 some elliptic curve group, 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 user and verifier when running the ShowCredential protocol consists of elements of the following formal language:

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

where σ ranges over the set of public keys of the credential scheme, and where n is the amount of attributes of σ. 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 $w = ((k'_0, \ldots, k'_n), s)$ 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 user has a valid credential then it can make the verifier accept.*

Proof. If the user 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

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

so the user can perform this proof without problem. □

4.1 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 user, 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 $\bar{K}, \bar{S}, (\bar{S}_i)_{i=0,\dots,n}, \tilde{T}$ that the user 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 $(k_0, \dots, k_n), (\kappa, \bar{K}, \bar{S}, \bar{S}_0, \dots, \bar{S}_n, \bar{T})$.

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 user $\mathcal{P}^*$ 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 $(\hat{K}, \hat{S}, \hat{S}_0, \dots, \hat{S}_n) \in G_1^{n+3}$ 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).[5] 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.

The first thing to notice is that if we take a setup of $n = 0$ attributes, then credentials in this setup are LRSW instances. The public key of such a setup would be $Q, \bar{A} = Q^{\bar{a}}, Z = Q^z$ (where the reason for the bar on top of the second group element will become clear shortly), and a valid credential would be $\bar{\kappa}, K, \bar{S} = K^{\bar{a}}, T = (K\bar{S}^{\bar{\kappa}})^z = K^{z+\bar{\kappa}\bar{a}z}$, which is indeed an LRSW instance. Suppose that algorithm $\mathcal{B}$ is given a list of $q = q_I + q_S$ such LRSW-instances

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

along with $Q, \bar{A}, Z$. Algorithm $\mathcal{B}$, acting as the challenger, engages in the unforgeability game with $\mathcal{A}$, acting as follows.

[5] 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.

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 public key

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

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 the LRSW-instance $\bar{\kappa}_j, K_j, \bar{S}_j, T_j$, note that the challenger can compute elements $S_{i,j}$ which are valid with respect to its public key 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 normally, but when the adversary proves knowledge of its private key $k_{0,j}$ and the number κ', $\mathcal{B}$ extracts 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 $\kappa_j = (\bar{\kappa}_j - \sum_{i=0}^n k_{i,j} a_i)/a$), and uses the number $\kappa_j - \kappa'$ as the value for κ'' in the remainder of the protocol.

Notice that we then have

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

and

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

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.

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 LRSW-instances as it does in Issue queries, randomly 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, 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

$$T = \Big(KS^{\kappa}\prod_{i=0}^{n} S_i^{k_i}\Big)^{z} = \Big(K\bar{S}^{\kappa a + \sum_{i=0}^{n} k_i a_i}\Big)^{z} = \big(K\bar{S}^{\bar{\kappa}}\big)^{z} = K^{z+\bar{\kappa}\bar{a}z}.$$

This implies that the tuple $\bar{\kappa}$, K, $\bar{S}$, T is a valid LRSW-instance. In order to derive a contradiction with the q-whLRSW assumption, it remains to show that with non-negligible 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 $\mathsf{ShowCredential}$ 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, otherwise the output of $\mathcal{A}$ would not have been a new credential. On the other hand, if j was a $\mathsf{ShowCredential}$ 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 with overwhelming probability. 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 $(K, S, S_0, \dots, S_n)$ 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 $i = 0, \dots, n$. 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 LRSW-instance, contradicting the q-whLRSW assumption. □

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

Proof (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 $(k_0, \dots, k_n)$, $\mathcal{B}$ 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 $\mathcal{A}$. □

4.2 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 user. 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 $\hat{K}, \hat{S}, \hat{S}_0, \dots, \hat{S}_n$ 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 $\hat{K}, \hat{S}, \hat{S}_0, \dots, \hat{S}_n, \hat{C}, \hat{T}$ as an extra credential of which the numbers $(\kappa, k_0, \dots, k_n)$ 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 user. 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 users, to any verifier that is given any auxiliary information. First notice that for honest users and the simulator alike, the elements $\bar{K}$ and $\tilde{C}$ are always randomly distributed in G_1. Also, again for both honest users 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 user $\mathcal{P}$. □

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 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 (9), 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. □

4.3 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. If a user has a credential valid with respect to σ and another one valid with respect to σ' and the credentials have the same secret key, then the user can simultaneously disclose attributes from both credentials, in such a way that the verifier is assured that the two credentials are owned by one user, as opposed to two colliding users. This works as follows.

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

$$w = ((k_0, k_1, \dots, k_n), (K, S, S_0, \dots, S_n, T)),$$
$$w' = ((k_0, k'_1, \dots, k'_n), (K', S', S'_0, \dots, S'_n, T')),$$

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

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

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 user can control the private key during the issuing process, and this is indeed the case (see Figure 1). To ensure that each user 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

$$T = \Big(K S^{\kappa''} S_0^{k''_0} R \prod_{i=1}^{n} S_k^{k_i} \Big)^z$$

to the user 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 user 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 user 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 user and issuer perform the Issue protocol as in Figure 1. However, the user combines the proof of knowledge over the hidden attributes with the proof of knowledge over k_0 and κ in the Issue protocol, additionally showing that k_0 coincides with the secret key from the credential that it showed.

5 Performance

5.1 The Fiat-Shamir heuristic

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 [6,7,21] to the Schnorr Σ-protocol [36] for DL-representations as follows: the user receives a nonce $\eta \in_R \mathbb{Z}_p^*$ from the verifier, and uses the Schnorr Σ-protocol with $c = H(W, D, \eta)$ as the challenge in the second move (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 would not only lower the amount of exponentiations for the user in the ShowCredential protocol, but also reduce the amount of moves to just two: after receiving the nonce η from the verifier, the user can combine the elements $\bar{K}, \bar{S}, \bar{S}_0, \ldots, \bar{S}_n, \tilde{C}, \tilde{C}$ and the proof of knowledge over η in a single message to the verifier (such a message is called a *disclosure 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 [17,24].

5.2 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 user has to perform. In this section we will therefore count the number of exponentiations the user has to perform.

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 user should use the right hand side to determine D. Moreover, if the user stores the elements $R := S^{\kappa}$ and $R_i := S_i^{k_i}$ for $i = 0, \ldots, n$, then it can calculate D by

$$D = \Big(K \prod_{i \in \mathcal{D}} R_i \Big)^{-\alpha} = \Big(C^{-1} R R_0 \prod_{i \in \mathcal{C}} R_i \Big)^{\alpha} \tag{7}$$

both of which take just one exponentiation.

Denote with $\mathrm{pk}(i)$ the amount of exponentiations the user has to compute in the zero-knowledge proof of knowledge when it presents a DL-representation of length i. Then the number of exponentiations in G_1 that the user has to do is

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

The user 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).

Table 1 compares the amount of exponentiations in our scheme to those of [15], U-Prove and Idemix. However, note that exponentiations in RSA-like groups, on which Idemix depends, are significantly more expensive than exponentiations in elliptic curves. Also, the U-Prove showing protocol offers no unlinkability. As to the scheme from [15], 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 user has to do if he voluntarily discloses some attributes. We see that the amount of exponentiations that the user has to perform in the ShowCredential protocol of [15] 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.

5.3 Implementation

In order to further examine the efficiency of our credential scheme we have written a preliminary implementation, using the high-speed 254-bit BN-curve and pairing implementation from [8]. The latter is written in C++ and assembly but also offers a Java API, and it uses the GMP library from the GNU project[6] for large integer arithmetic. Table 2 shows the running times of our implementation along with those from the Idemix implementation from the IRMA project.[7] We have tried to make the comparison as honest as possible by writing our implementation in Java and using the Fiat-Shamir heuristic, like the IRMA Idemix implementation, which we have modified to also use the GMP library for its large integer arithmetic. However, the comparison can still only go so far, because the elliptic curve group that [8] offers is heavily optimized for fast computations, from which our scheme profits because it allows multiple issuers to use the same group. Such optimizations are not possible in Idemix because each Idemix public key necessarily involves its own group. Moreover, the IRMA Idemix implementation is 1024-bits, which according to [30] corresponds to a 144 bit curve (see also `www.keylength.com`), so that the two implementations do not offer the same level of security.

For these reasons we will go no further than draw qualitative conclusions from the data. Nevertheless, both remarks actually demonstrate the efficiency of our scheme: the first means that our scheme can be optimized further than Idemix could, and Table 2 shows that even though our implementation offers a much higher level of security, it is still significantly faster than the IRMA Idemix implementation. We believe therefore that the conclusion that our scheme is or can be more efficient than Idemix – at least for the user in the ShowCredential protocol – is justified. Apart from this, the table also highlights the following differences between the two:

[6] See `gmplib.org`.
[7] See `irmacard.org` and `github.com/credentials`.

Table 1. Exponentiation and pairing count for the user 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 user has to compute. The number n denotes the amount of attributes, excluding the secret key, and the function $\mathrm{pk}(n)$ denotes the amount of exponentiations necessary in order to perform a zero-knowledge proof of knowledge of n numbers (in the case of the Fiat-Shamir heuristic applied to the Schnorr Σ-protocol, which Idemix also uses, we have $\mathrm{pk}(n) = n$). In the case of our own scheme, we assume that the user calculates D in the ShowCredential protocol using the elements R_i as in equation (7).

	G_{EC}	G_T	e	G_{RSA}	unlinkable
Our scheme	$n + \mathrm{pk}(\lvert\mathcal{C}\rvert + 3) + 6$	0	0	0	yes
[15]	$2n + 3$	$\mathrm{pk}(\lvert\mathcal{C}\rvert + 2)$	$n + 3$	0	yes
Idemix	0	0	0	$\lvert\mathcal{C}\rvert + 3$	yes
U-Prove	$\lvert\mathcal{C}\rvert + 1$	0	0	0	no

Table 2. A comparison of the running times of various actions in the implementation of our credential scheme and the IRMA Idemix implementation, both of them using the Fiat-Shamir heuristic. The columns labeled "computing proof" and "verifying proof" show how long it takes to compute and to verify a disclosure proof, respectively, while the column labeled "verifying credential" shows how long it takes to verify the signature of a credential. The left column shows the total number of attributes and, if applicable, the amount of disclosed attributes (this does not apply to the "verifying credential" column). In the case of our own scheme, we let the user calculate D in the ShowCredential protocol using the elements R_i as in equation (7). The attributes were randomly chosen 253-bit integers, the same across all tests, and the computations were performed on a dual-core 2.7 GHz Intel Core i5. All running times are in milliseconds, and were obtained by computing the average running time of 1000 iterations.

# attributes	computing proof		verifying proof		verifying credential	
total (discl.)	This work	Idemix	This work	Idemix	This work	Idemix
6 (1)	2.9	11.7	5.7	11.2	5.1	6.5
7 (1)	2.9	12.6	6.5	12.2	5.8	6.9
8 (1)	3.2	13.4	7.1	13.2	6.6	7.4
9 (1)	3.4	14.3	8.0	14.0	7.2	7.7
10 (1)	3.7	15.2	8.7	14.9	7.8	8.3
11 (1)	3.9	16.5	9.4	15.8	8.6	8.7
12 (1)	4.2	17.1	10.2	16.9	9.0	8.9
6 (5)	2.1	7.6	5.9	9.2		
7 (6)	2.1	7.5	6.5	9.7		
8 (7)	2.3	7.5	7.2	10.1		
9 (8)	2.4	7.4	7.9	10.7		
10 (9)	2.6	7.4	8.5	10.9		
11 (10)	2.7	7.5	9.1	11.4		
12 (11)	2.8	7.5	9.9	12.0		

- In our scheme, verifying the validity of a disclosure proof tends to be two or three times as expensive as creating one, as it involves calculating a number of pairings. By contrast, in Idemix, verifying a disclosure proof is about as expensive as creating one.
- Verifying the validity of a credential in Idemix is significantly cheaper than computing or verifying disclosure proofs; while in our scheme verifying credential validity is only slightly cheaper than verifying disclosure proofs (again, because of having to compute pairings).
- When all but one attributes are disclosed (so that $|\mathcal{C}| = 1$ is constant), the ShowCredential protocol of our scheme becomes noticeably cheaper for the user. In Idemix, however, the user cost stops growing at all as the total amount of attributes increases. Referring to Table 1, this is because the amount of exponentiations in our scheme depends on n, while it does not in the case of Idemix. Verifying disclosure proofs becomes slightly cheaper in our scheme, but not by much because the amount of pairing stays the same; here too the differences are more pronounced in Idemix.

6 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), on a comparison of exponentiation counts, and on a comparison of run times with the IRMA Idemix implementation, we have shown it to be more efficient than comparable schemes such as Idemix and the scheme from [15], achieving the same security goals at less cost.

References

1. Antipa, A., Brown, D., Menezes, A., Struik, R., Vanstone, S.: Validation of elliptic curve public keys. In: Desmedt, Y. (ed.) Public Key Cryptography – PKC 2003. LNCS, vol. 2567, pp. 211–223. Springer Berlin Heidelberg (2002)
2. Ateniese, G., Camenisch, J., de Medeiros, B.: Untraceable rfid tags via insubvertible encryption. In: Proceedings of the 12th ACM Conference on Computer and Communications Security - CCS '05. pp. 92–101. ACM, New York, NY, USA (2005)
3. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. pp. 1087–1098. CCS '13, ACM, New York, NY, USA (2013)
4. Baldimtsi, F., Lysyanskaya, A.: On the security of one-witness blind signature schemes. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013. LNCS, vol. 8270, pp. 82–99. Springer Berlin Heidelberg (2013)
5. Barreto, P., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) Selected Areas in Cryptography. LNCS, vol. 3897, pp. 319–331. Springer Berlin Heidelberg (2006)

6. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security. pp. 62–73 (1993)
7. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. LNCS, vol. 7658, pp. 626–643. Springer Berlin Heidelberg (2012)
8. Beuchat, J., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barreto-naehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing-Based Cryptography - Pairing 2010. Lecture Notes in Computer Science, vol. 6487, pp. 21–39. Springer (2010)
9. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: 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. pp. 326–349. ACM, New York, NY, USA (2012)
10. Blake, I.F., Seroussi, G., Smart, N.P. (eds.): Advances in Elliptic Curve Cryptography. Cambridge University Press (2005), cambridge Books Online
11. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)
12. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press (2000)
13. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. In: Naor, M. (ed.) Advances in Cryptology - EUROCRYPT 2007. pp. 246–263. Springer Berlin Heidelberg, Berlin, Heidelberg (2007), `https://eprint.iacr.org/2007/172.pdf`
14. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer Berlin Heidelberg (2001)
15. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer Berlin Heidelberg (2004)
16. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski, B.S.J. (ed.) Advances in Cryptology — CRYPTO '97. LNCS, vol. 1294, pp. 410–424. Springer Berlin Heidelberg (1997)
17. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
18. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) Advances in Cryptology - CRYPTO '92. LNCS, vol. 740, pp. 89–105. Springer (1993)
19. Cramer, R., Damgård, I., MacKenzie, P.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Imai, H., Zheng, Y. (eds.) Public Key Cryptography. LNCS, vol. 1751, pp. 354–372. Springer Berlin Heidelberg (2000)
20. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91. LNCS, vol. 576, pp. 445–456. Springer (1991)
21. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology – CRYPTO' 86. LNCS, vol. 263, pp. 186–194. Springer Berlin Heidelberg (1987)

22. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
23. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, New York, NY, USA (2000)
24. Goldwasser, S., Kalai, Y.T.: On the (in)security of the fiat-shamir paradigm. In: 44th Symposium on Foundations of Computer Science - FOCS 2003. pp. 102–113. IEEE Computer Society (2003)
25. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
26. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (apr 1988)
27. Hajny, J., Malina, L.: Unlinkable attribute-based credentials with practical revocation on smart-cards. In: Mangard, S. (ed.) Smart Card Research and Advanced Applications. LNCS, vol. 7771, pp. 62–76. Springer Berlin Heidelberg (2013)
28. Hanzlik, L., Kluczniak, K.: A short paper on how to improve U-Prove using self-blindable certificates. In: Christin, N., Safavi-Naini, R. (eds.) Financial Cryptography and Data Security: 18th International Conference, FC 2014. pp. 273–282. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
29. IBM Research Zürich Security Team: Specification of the Identity Mixer cryptographic library, version 2.3.0. Tech. rep., IBM Research, Zürich (feb 2012), `https://tinyurl.com/idemix-spec`
30. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. J. Cryptology 14(4), 255–293 (2001)
31. Lysyanskaya, A.: Pseudonym Systems. Master's thesis, Massachusetts Institude of Technology (1999), `https://groups.csail.mit.edu/cis/theses/anna-sm.pdf`
32. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H., Adams, C. (eds.) Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99. pp. 184–199. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
33. Miyaji, A., Nakabayashi, M., Takano, S.: 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)
34. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 96–109. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
35. Paquin, C., Zaverucha, G.: U-prove cryptographic specification v1.1 (revision 3) (December 2013), `http://research.microsoft.com/apps/pubs/default.aspx?id=166969`, released under the Open Specification Promise
36. Schnorr, C.: Efficient identification and signatures for smart cards. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT '89. LNCS, vol. 434, pp. 688–689. Springer Berlin Heidelberg (1990)
37. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) Advances in Cryptology — EUROCRYPT '97. LNCS, vol. 1233, pp. 256–266. Springer Berlin Heidelberg (1997)
38. Verheul, E., Ringers, S., Hoepman, J.H.: The self-blindable U-Prove scheme from FC'14 is forgeable. Financial Cryptography and Data Security – FC'16 (2016), `https://eprint.iacr.org/2015/725`
39. Verheul, E.R.: Self-blindable credential certificates from the weil pairing. In: Boyd, C. (ed.) Advances in Cryptology - ASIACRYPT. LNCS, vol. 2248, pp. 533–551. Springer (2001)

40. Vullers, P., Alpár, G.: Efficient selective disclosure on smart cards using idemix. In: Fischer-Hübner, S., de Leeuw, E., Mitchell, C. (eds.) Policies and Research in Identity Management. IFIP Advances in Information and Communication Technology, vol. 396, pp. 53–67. Springer Berlin Heidelberg (2013)
41. Wachsmann, C., Chen, L., Dietrich, K., Löhr, H., Sadeghi, A.R., Winter, J.: Lightweight anonymous authentication with tls and daa for embedded mobile devices. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) Information Security: 13th International Conference, ISC 2010. pp. 84–98. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), https://eprint.iacr.org/2011/101.pdf
42. Wei, V.K., Yuen, T.H.: More short signatures without random oracles. IACR Cryptology ePrint Archive 2005, 463 (2005), 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 $K_1, \dots, K_m \in G$ all be distinct. If for some $K \in G$ the numbers $(k_1, \dots, k_m) \in \mathbb{Z}_p^m$ are such that $K = \prod_{j=1}^m K_j^{k_j}$, then $(k_1, \dots, k_m)$ is called a *DL-representation* of K with respect to $(K_1, \dots, K_m)$. 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, [12, p. 60].

Proposition 20. *Let $(K_1, \dots, K_m)$ be randomly distributed. No probabilistic polynomial-time algorithm can, on input $(K_1, \dots, K_m)$, generate a non-trivial DL-representation of $1 \in G$ with respect to $(K_1, \dots, K_m)$ 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 $X_1, \dots, X_m \in G_1$ and the tuple $Y_1, \dots, Y_m$ 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 $X_1, \dots, X_m, Y_1, \dots, Y_m$ generate a non-trivial DL-representation of $1 \in G_1$ with respect to $(X_1, \dots, X_m)$.*

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 [11], 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 $\text{view}_{\mathcal{A}}(\mathcal{A}(x, a) \leftrightarrow \mathcal{B}(x, w))$ 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 ([25], see also [19,23]) 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 $R(x, w) = 1$,

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

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

$$\begin{aligned} &\{\text{view}_{\mathcal{V}^*}(\mathcal{P}(x, w) \leftrightarrow \mathcal{V}^*(x, a))\}_{x \in L,\, a \in \{0,1\}^*} \\ &\qquad \overset{c}{\approx} \{\text{view}_{\mathcal{V}^*}(\mathcal{S}(x) \xrightarrow{\blacksquare} \mathcal{V}^*(x, a))\}_{x \in L,\, a \in \{0,1\}^*} \end{aligned} \tag{8}$$

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

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

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

$$\Pr[\mathcal{P}^*(x, a) \xleftarrow{\blacksquare} \chi(x) \rightarrow w : R(x, w) = 1] \geq \epsilon(|x|) - \nu(|x|).$$

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:

$$\left| \Pr[\mathcal{P}(x, w) \leftrightarrow \mathcal{V}^*(x, a) \rightarrow 1] - \Pr[\mathcal{S}(x) \xrightarrow{\blacksquare} \mathcal{V}^*(x, a) \rightarrow 1] \right| < \text{negl}(|x|). \tag{9}$$

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 (8).

Definition 23. We define existential unforgeability of a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ under adaptive chosen message attacks in terms of the following game [26]. 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 $m_1, \ldots, m_q$ 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 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 $\mathsf{ShowCredential}$ protocol with the adversary on credential j_b, acting as the user. 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 $j \in_R \{1,\ldots,m\}$, 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.