Efficient self-blindable attribute-based credentials

Sietse Ringers[1], Eric Verheul[2], and Jaap-Henk Hoepman[2]

[1] Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, The Netherlands
s.ringers@rug.nl
[2] Radboud University, Nijmegen, The Netherlands
{jhh,e.verheul}@cs.ru.nl

Abstract. An attribute-based credential 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 are additionally unlinkable: that is, if the same attributes were disclosed in two transactions, it is not possible to tell if one or two credentials were involved. However, to our knowledge no such scheme is known that is sufficiently efficient to allow usable implementations on smart cards. In this paper we present a new attribute-based credential scheme, of which we prove that it is unforgeable and unlinkable. Defined on elliptic curves, the scheme involves bilinear pairings but only on the verifier's side, allowing for efficient implementations on smart cards.

1 Introduction

An attribute-based credential 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 have already been proposed, of which some provide *unlinkability*, i.e., the verifier cannot tell whether two transactions did or did not originate from the same user (assuming the same attributes were disclosed in both transactions).

Two well-known schemes are Idemix [CL01; IBM12] and U-Prove [Bra00; PZ13].[3] However, to date there is no provably secure scheme that is sufficiently efficient to safely allow implementations on smart cards (Idemix, being based on the strong RSA-problem, requires bit lengths of 2048 bits or more), while also providing unlinkability of transactions (U-Prove is more efficient but does not provide unlinkability; in addition, its security is not fully proven).

In this paper, we provide a new provably secure, efficient and unlinkable attribute-based credential scheme, as follows. We take the attributes to be elements of $\mathbb{Z}_p$ for a certain prime number p, and we define a signature scheme on the space of attributes $\mathbb{Z}_p^n$. These signatures will include a *secret key* k_0, of which we will sometimes think as the zeroth attribute. A credential is then a set of attributes $(k_0, \ldots, k_n)$, together with a signature on this set of attributes.

Next, we provide two interactive protocols:

Issue An interactive protocol between the issuer $\mathcal{I}$ and a prover $\mathcal{P}$, endowing the prover $\mathcal{P}$ with a credential containing certain attributes on which the issuer and prover agreed previously. This protocol should be such that the issuer does not learn the secret key k_0 of the prover's credential.

ShowCredential An interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$, in which the prover first sends a number of attributes to the verifier, and afterwards proves to the verifier that it possesses a credential containing those attributes.[4]

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

[4] As in Idemix and U-Prove, our ShowCredential protocol can be extend to simultaneously show multiple credentials that have the same secret key (see Section 4.1) or arbitrary linear combinations [Bra00].

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

- *Unforgeability* (see Definition 2): Only the issuer can create valid credentials, and valid credentials cannot be modified without invalidating the credential's signature.
- *Multi-show unlinkability* (see Definition 3): If a verifier $\mathcal{V}$ executes the ShowCredential protocol 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 two different ones.
- *Issuer unlinkability*: The issuer cannot decide if a run of the Issue and a run of the ShowCredential protocol did or did not originate from the same credential.
- *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.

The unforgeability of our credential scheme will depend on the Known Exponent Assumption (KEA), a powerful assumption that is seeing more and more use in recent literature. After having defined attribute-based credential schemes as well as unforgeability and unlinkability in the next session, we will discuss this assumption in Section 3. In the same section we will introduce and prove unforgeability of a signature scheme on the space of attributes, that our credential scheme will use. In section 4 we turn to our credential scheme, defining issuing and showing protocols, and prove that these provide unforgeability and unlinkability. In Sections 5 and 6 we will discuss the performance of our scheme, and compare it to a number of other attribute-credential schemes.

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. 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 algorithm $\mathcal{B}$ – that is, $\mathcal{A}$ has oracle access to the next-message function function $\mathcal{B}_{x,y,r}(m)$, which, on common input x, 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 D) we will use the Camenisch-Stadler notation [CS97]: for example,

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

denotes a zero-knowledge proof of knowledge of the numbers k_1, k_2 that satisfy the relation $K = P_1^{k_1} P_2^{k_2}$. (We will, however, not switch to Greek letters to denote the variables of which knowledge is proved.)

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. It may be chosen by the prover and 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.

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

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

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

We define unlinkability and unforgeability 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 The adversary decides on the number of attributes $n \geq 1$ 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 challenger.

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

Challenge The challenger, now acting as the verifier, and the adversary, acting as the user, engage in the $\mathsf{ShowCredential}$ protocol. If the adversary manages to make the verifier accept a credential with disclosed attributes $(k_i)_{i \in \mathcal{D}}$, and there is no j such that $k_i = k_{i,j}$ for all $i \in \mathcal{D}$ (i.e., there is no credential 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 E).

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 The adversary decides on the number of attributes $n \geq 1$ 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 $(k_{1,j}, \ldots, k_{n,j})$.
- $\mathsf{ShowCredential}(j)$ 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 j_0, j_1 and $\mathcal{D}$ to the challenger, who checks that this holds; if it does not then the adversary lost.
Next, the challenger flips a bit $b \in_R \{0, 1\}$, and acting as the prover, 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*.

Theorem 4. *Let* $(\mathsf{KeyGen}, \mathsf{Issue}, \mathsf{ShowCredential})$ *be an attribute-based credential scheme whose* $\mathsf{ShowCredential}$ *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 $\mathsf{ShowCredential}$ 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 $\mathcal{S}$).

3 Preliminaries

With $e\colon G_1 \times G_2 \to G_T$ we denote a Type 3 bilinear group pair (see Definition 15); we write p for the prime order of the three groups. 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 provers. 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 $\nu(k) < \mathrm{negl}(k)$ when the function $\nu\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligible; that is, for any polynomial p there exists a k such that $\nu(k) < \frac{1}{p(k)}$.

3.1 The XKEA assumption

The unforgeability of the credential schemes defined in the paper will depend on the Knowledge of Exponent assumption, or KEA for short (as well as the discrete logarithm assumption, defined in Appendix A). Let G be a cyclic group of prime order p, and let $1 \neq K \in G$. If an algorithm $\mathcal{A}$ is given K, K^a for some number $a \in \mathbb{Z}_p$, then it can generate a pair L, L^a by raising K and K^a to any power, i.e., by setting $(L, L^a) = (K^c, (K^a)^c)$ for any $c \in \mathbb{Z}_p$. The KEA assumption essentially states that this is the only way in which an algorithm can generate such a pair. There are many versions of this assumption, differing in how many such pairs the algorithm gets as input; if and how they are related; and if the algorithm is allowed to get auxiliary input besides these pairs.[5] The following version is the original one, introduced by Darmgård in [Dam91].

[5] There are also differences in the names given to the assumption: the KEA assumption is sometimes also called KEA1, SDHA-1 (for strong Diffie-Hellman), the KE assumption, or DHK (for Diffie-Helmann

Definition 5 (KEA assumption). Let G be a cyclic group, whose prime order p is k bits, and let $1 \neq K \in G$ and $a \in \mathbb{Z}_p^*$. The Knowledge of Exponent (KEA) assumption holds in G, if for any probabilistic polynomial-time algorithm $\mathcal{A}$ that on input K, K^a gives as output $L, L' \in G$, there exists a probabilistic polynomial-time algorithm $\chi_{\mathcal{A}}$ called the *extractor* that, when given the same input and random coins of $\mathcal{A}$ returns $c \in \mathbb{Z}_p$ which is such that

$$\Pr[L' = L^a \ \wedge \ L \neq K^c] < \text{negl}(k).$$

The wording of the assumption leaves open the possibility that the output of $\mathcal{A}$ does not have a as the relative discrete log (for example, $\mathcal{A}$ may output that it failed). When it does, however, output a pair having a as relative discrete log, then the probability that the extractor does not manage to find the discrete log of L with respect to K is negligible.

The KEA assumption has been criticized because it seems harder to disprove it than assumptions such as the discrete logarithm-assumptions. On the other hand, the KEA assumption can be proven to hold in the *generic group model*, in which the adversary is oblivious of the actual representation of the group elements and only performs generic group operations such as multiplying, inverting, and testing for equality. In for example [Gro10] this is proved even in the presence of bilinear pairings. The assumption and its various generalizations are used in areas such as (non-interactive) zero-knowledge proofs [Gro10; Bit+14; Bit+12; BP04a], encryption [BP04b; WS08; Dam91], 2-party computation [DFH12], authentication and key-exchange [DG09; DGK06; Kra05; YZ10; WS07].

The version we will use follows, for example, [Bit+12; WS07]. It generalizes the assumption above by allowing $\mathcal{A}$ to receive more than one pairs of the form (K, K^a), and to a Type 3 pairing setting. For more details on the XKEA assumption and the generic group model we refer to Appendix C.

Definition 6 (XKEA assumption). Let $G_1 \times G_2 \to G_2$ be a bilinear pairing, where the order p of the three groups is k bits, and let and $a \in \mathbb{Z}_p^*$. Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm taking as input $K_1, \ldots, K_m$, $K_1^a, \ldots, K_m^a \in G_1$, $Q_1, \ldots, Q_r \in G_2$, where

- m and r are polynomial in k,
- $K_j \neq 1$ for all $j = 1, \ldots, n$.

The Extended Knowledge of Exponent (XKEA) assumption holds in G, if when $\mathcal{A}$ gives as output L, L' then there exists a probabilistic polynomial-time extractor $\chi_{\mathcal{A}}$ that, when given the same input and random coins of $\mathcal{A}$ returns a tuple $c_1, \ldots, c_n$ which is such that

$$\Pr[L' = L^a \ \wedge \ L \neq K_1^{c_1} \cdots K_m^{c_m}] < \text{negl}(k).$$

We will sometimes also write $\chi_{\mathcal{A},a}$ to indicate the number a with respect to which we apply the assumption.

3.2 A signature scheme on the space of attributes

The Chaum-Pedersen signature scheme [CP93] serves as the basis for our signature scheme on $\mathbb{Z}_p^n$, 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. A signature on a message $K \in G_1$ is $S = K^a$, and it is verified by $e(S, Q) \stackrel{?}{=} e(K, A)$.

Knowledge). Meanwhile, the XKEA assumption is sometimes referred to as GKEA (for Generalized KEA), or n-KEA, or KEA3 when $n = 2$. The more recent papers seem to have settled on some variant of KEA.

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 7 (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 $T = (KS^\kappa \prod_{i=0}^n S_n^{k_n})^z$. The signature is $(\kappa, K, S, S_0, \ldots, S_n, T)$.

Verify$((k_0, \ldots, k_n), (\kappa, K, S_0, \ldots, S_n, T), \sigma)$ The signature is checked by setting $C = KS^\kappa \prod_{i=0}^n S_i^{k_i}$ and verifying

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

We call the numbers $k_n \in \mathbb{Z}_p$ 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 $\{0, \ldots, 2^{\ell-1}\}$. 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.

The element $C = KS^\kappa \prod_{i=0}^n S_n^{k_n}$ is called the *attribute commitment* of the attributes $k_0, \ldots, k_n$. Although the attribute commitment C 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 an element of the form $(k_0, \ldots, k_n), (\kappa, K, S_0, \ldots, S_n, T)$ where $(\kappa, K, S_0, \ldots, S_n, T)$ is a valid signature over $(k_0, \ldots, k_n)$ a *credential*.

Notice that if $(k_0, \ldots, k_n), (\kappa, K, S_0, \ldots, S_n, T)$ is a valid credential, then $(k_0, \ldots, k_n), (\kappa, K^\alpha, S_0^\alpha, \ldots, S_n^\alpha, T^\alpha)$ for $\alpha \in \mathbb{Z}_p^*$ is another valid credential having the same attributes. That is, in the terminology of [Ver01] our credentials are *self-blindable*. At the same time, it is not clear how to change the credential into another one over different attributes, and indeed, we will prove in Theorem 9 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 scheme (Theorem 9).[6]

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 $a_{n+1}, \ldots, a_{n'} \in_R \mathbb{Z}_p^*$, and adding the elements $A_{n+1} = Q^{a_{n+1}}, \ldots, A_{n'} =$

[6] We could have eased the notation somewhat by denoting the randomizer κ as an extra attribute k_{n+1}, but because it plays a rather different role than the other attributes (it is chosen by the issuer, and part of the signature), we believe this would create more confusion than ease. On the other hand, notice that if κ *were* considered part of the message that is signed, then Lemma 8 says that our signature scheme is invulnerable under known-message attacks.

$Q^{a_{n'}}$ 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 $k_{n+1}, \ldots, k_{n'}$ will be 0.

In the remainder of this section we prove unforgeability of this signature scheme. In the lemma below, we take for the moment $n = 0$; that is, there are no attributes and our credentials are of the form $(\kappa_j, K_j, S_j, T_j)$. The attribute commitment of such a credential is $C_j = K_j S_j^{\kappa_j}$, and it is valid only if $T_j = C_j^z$ and $S_j = K_j^a$. Although in practice it would not make much sense to create credentials without attributes in this way, we now show that only the issuer can create these; later (in Theorem 9) we will reduce the unforgeability of the full signature scheme (with $n > 0$) to this lemma.

Lemma 8. *Consider a list of m credentials $(\kappa_j, K_j, S_j, T_j)_{j=1,\ldots,m}$, where the group elements K_j are randomly distributed. Given such a list, no probabilistic polynomial-time algorithm can output a new valid credential (κ, K, S, T) such that $\kappa \neq \kappa_j$ for all j.*

Proof. Suppose we have an algorithm $\mathcal{A}$ that violates the statement above. First we wrap the algorithm $\mathcal{A}$ in another algorithm $\mathcal{B}$ to which the XKEA assumption is applicable; see Algorithm 1. Essentially this is the same algorithm, but with its input and output parameters such that the XKEA assumption can be applied to it. (In particular, the C_j present in the parameter list of $\mathcal{B}$ serve no purpose other than allowing us to use the XKEA assumption.) After having extracted a number of coefficients from $\mathcal{B}$ using the extractor from the XKEA assumption, we will derive a non-trivial DL-representation of 1, which is impossible because of the DL-assumption (see Proposition 20 in Appendix B).

Algorithm 1 The algorithm $\mathcal{B}$.

function $\mathcal{B}\Big((\kappa_j, K_j, S_j, C_j, T_j)_{j=1,\ldots,m}\Big)$
  for all $j \in \{1, \ldots, m\}$ **do**
    if $C_j \neq K_j S_j^{\kappa_j}$ **then**
      return $\perp$
    end if
  end for
  $(\kappa, K, S, T)) \leftarrow \mathcal{A}\Big((\kappa_j, K_j, S_j, T_j)_{j=1,\ldots,m}\Big)$
  $C \leftarrow KS^{\kappa}$
  return K, S, C, T
end function

We can interpret $(K_j, S_j = K_j^a)$ of the j-th credential as a Chaum-Pedersen signature over K_j with secret key a. This means that if $K = \prod_{j=1}^m K_j^{c_j}$, then we must also have $S = \prod_{j=1}^m S_j^{c_j}$ for the same numbers c_j. These can be extracted using the XKEA extractor, considering all other input to $\mathcal{B}$ to be auxiliary.[7]

Furthermore, again using the XKEA extractor on C and T (and considering the other input to be auxiliary), we obtain a set of numbers $d_1, \ldots, d_m$ such that

$$T = \prod_{j=1}^{m} T_j^{d_j}, \qquad C = \prod_{j=1}^{m} C_j^{d_j} = \prod_{j=1}^{m} K_j^{d_j} S_j^{\kappa_j d_j}.$$

[7] There is a subtlety here, namely that the auxiliary input is more than just elements of G_2. We will address this issue in Appendix C, by providing an explicit reduction to the XKEA assumption as in Definition 6.

On the other hand, C should be equal to

$$C = KS^{\kappa} = \prod_{j=1}^{m} K_j^{c_j} S_j^{\kappa c_j}.$$

Comparing the right hand sides of the two equations gives

$$\prod_{j=1}^{m} K_j^{d_j} S_j^{\kappa_j d_j} = \prod_{j=1}^{m} K_j^{c_j} S_j^{\kappa c_j}$$

This results in a DL-representation of 1, namely

$$1 = \prod_{j=1}^{m} K_j^{d_j - c_j} S_j^{\kappa_j d_j - \kappa c_j}. \tag{1}$$

Next, we show that the numbers

$$d_j - c_j \qquad \kappa_j d_j - \kappa c_j \tag{2}$$

are not all equal to 0, so that (1) is a nontrivial representation of 1. Suppose they do all equal 0. This gives the following set of equations for all j:

$$d_j = c_j, \qquad \kappa c_j = \kappa_j d_j. \tag{3}$$

These equations imply that for each j, we either have $(c_j, d_j) = (0, 0)$, or

$$\kappa = \kappa_j.$$

There must exist at least one pair (c_j, d_j) unequal to $(0, 0)$, otherwise we would have $C = 1$ which is not valid. Therefore, the equation above contradicts the assumption that the adversary outputted a credential with a new κ.

Thus the numbers (2) do not all equal 0, so that the DL-representation of 1 in equation (1) is nontrivial.

We remark that the lemma would remain true if we were to restore the presence of the attributes $(k_{0,j}, \ldots, k_{n,j})$. Equation (3) would then be augmented by $k_{i,j} d_j = k_i c_j$ for each i, and the remainder of the proof would work as is. The lemma would then essentially say that our credentials are unforgeable under known-message attacks. We have not done this for simplicity, and because the lemma in the form above already allows us to prove unforgeability under adaptively chosen message-attacks (from which unforgeability under known-message attacks follows).

Theorem 9. *Our credentials are existentially unforgeable under adaptively chosen message attacks.*

Proof. Suppose $\mathcal{A}$ can win the unforgeability game (see Definition 18) with non-negligible advantage. We build an algorithm $\mathcal{B}$ that contradicts Lemma 8 above.

Consider a setup with only a zeroth attribute, $Q, Z, \bar{A} = Q^{\bar{a}}$, where the reason for the bar on top of the last group element will become clear shortly. We will refer to this setup as the original setup. Suppose that algorithm $\mathcal{B}$ is given a list of credentials

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

that is valid with respect to this setup. $\mathcal{B}$, 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 In the j-th query to $\mathcal{B}$, $\mathcal{A}$ chooses a set of attributes $(k_{0,j}, \ldots, k_{n,j})$ and sends it to $\mathcal{B}$, who then 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 $\kappa_j = (\sum_{i=0}^{n} k_{i,j} a_i - \bar{\kappa}_j)/a$). Next, it sets

$$S_j = \bar{S}_j^a \qquad \text{and} \qquad S_{i,j} = \bar{S}_j^{a_i} \qquad \text{for all } i = 0, \ldots, m,$$

and then sends $\kappa_j, K_j, S_j, S_{0,j}, \ldots, S_{n,j}, T_j$ to $\mathcal{A}$.

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 with respect to the new setup.

Output $\mathcal{A}$ outputs a new 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

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

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 Lemma 8, we now show that with overwhelming probability $\bar{\kappa} \neq \bar{\kappa}_j$.

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

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

and since the credential output by $\mathcal{A}$ is new, these are two distinct DL-representation of the element $\bar{S}^{\bar{\kappa}}$. This results in a non-trvial DL-representation of 1, namely

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

This means that if we play the unforgeability game honestly with this adversary, that there is a non-negligible chance that we can find a non-trivial DL-representation of 1 in this way. As this would contradict Proposition 20, we conclude that we must have $\bar{\kappa} \neq \bar{\kappa}_j$ with overwhelming probability, which indeed contradicts Lemma 8. This completes the proof.

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 7), 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[8] is shown in Figure 1, and the ShowCredential protocol is shown in Figure 2. Here and in the remainder of the paper, we will write

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

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

Fig. 1. The Issue protocol.

Common information: Attributes $k_1, \ldots, k_n$, issuer's public key $\sigma = (p, e, Q, A, A_0, \ldots, A_n, Z)$	
Prover	**Issuer**
knows secret key k_0	knows $a, a_0, \ldots, a_n$, z
into K, S, S_0 $\longleftarrow$	send $K \in_R G_1, S = K^a, S_0 = K^{a_0}$
choose $\kappa' \in_R \mathbb{Z}_p^*$	
send $R = S^{\kappa'} S_0^{k_0}$ $\longrightarrow$	into R
$\mathrm{PK}\{(\kappa', k_0) \colon R = S^{\kappa'} S_0^{k_0}\}$ $\longleftrightarrow$	
	random $\kappa'' \in_R \mathbb{Z}_p$
	set $S_i = K^{a_i}$ $\forall i \in [1, n]$
	set $T = \left(K S^{\kappa''} R \prod_{i=1}^n S_i^{k_i}\right)^z$
into $S, S_1, \ldots, S_n, T, \kappa''$ $\longleftarrow$	send $S, S_1, \ldots, S_n, T, \kappa''$
set $\kappa = \kappa' + \kappa''$	
$\mathsf{Verify}((k_0, \ldots, k_n), (\kappa, K, S, S_0, \ldots, S_n, T), \sigma)$	
return $(k_0, \ldots, k_n), (\kappa, K, S, S_0, \ldots, S_n, T)$	

The Issue protocol is such that both parties contribute to κ, while the issuer is prevented from learning its value. Additionally, K is randomly distributed. This is important, because only then does Theorem 9 guarantee unforgeability. Since the issuer chooses K, it could decide to not choose them randomly, but this would only break the unforgeability of the system and not give him any advantage. Therefore it would not make sense for it to do so.

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 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.

[8] 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.

Fig. 2. The ShowCredential protocol.

Common information: Issuer's public key $\sigma = (p, e, Q, A, A_0, \ldots, A_n, Z)$; disclosure set $\mathcal{D} \not\ni 0$, undisclosed set $\mathcal{C} = \{1, \ldots, n\} \setminus \mathcal{D}$; disclosed attributes $(k_i)_{i\in\mathcal{D}}$

Prover		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 $\bar{K} = K^\alpha$, $\bar{S} = S^\alpha$		
set $\bar{S}_i = S_i^\alpha \; \forall i \in \{0, \ldots, n\}$		
set $\tilde{C} = C^{-\alpha/\beta}$, $\tilde{T} = T^{-\alpha/\beta}$		
send $\bar{K}$, $\bar{S}$, $(\bar{S}_i)_{i=0,\ldots,n}$, $\tilde{C}$, $\tilde{T}$	$\longrightarrow$	into $\bar{K}$, $\bar{S}$, $(\bar{S}_i)_{i=0,\ldots,n}$, $\tilde{C}$, $\tilde{T}$
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 $e(\bar{K}, A) \stackrel{?}{=} e(\bar{S}, Q)$
		verify $e(\bar{K}, A_i) \stackrel{?}{=} e(\bar{S}_i, Q) \; \forall i \in [0, n]$
		verify $e(\tilde{C}, Z) \stackrel{?}{=} e(\tilde{T}, Q)$

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:

$$L = \left\{ \left(\sigma, \mathcal{D}, (k_i)_{i\in\mathcal{D}}\right) \mid \mathcal{D} \subset \{1, \ldots, n\},\ k_i \in \mathbb{Z}_p \, \forall \, i \in \mathcal{D} \right\} \tag{5}$$

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 $w = ((k'_0, \ldots, k'_n), s)$ 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.) Then the equation $R(x, w) = 1$ holds only if w is a valid credential having attributes $(k_i)_{i\in\mathcal{D}}$.

In Theorems 10, 11 and 13 we will prove that our ShowCredential protocol is a black-box zero-knowledge proof of knowledge (see Definition 17) for this relation. From this fact unforgeability and unlinkability will follow.

Theorem 10. *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 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,

$$\tilde{C}^\beta \bar{S}^\kappa \bar{S}_0^{k_0} \prod_{i\in\mathcal{C}} \bar{S}_i^{k_i} = \bar{C}^{-1} \bar{S}^\kappa \bar{S}_0^{k_0} \prod_{i\in\mathcal{C}} \bar{S}_i^{k_i} = \bar{K}^{-1} \prod_{i\in\mathcal{D}} \bar{S}_i^{-k_i} = D, \tag{6}$$

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:

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

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:

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

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 Protocol 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 $k''_0 \in_R \mathbb{Z}_k*$, and then sets and sends

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

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 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 the proof is valid, the user and issuer perform the Issue protocol as in Figure 1. However, the user combines the two proofs of knowledge into one that uses the same secret key k_0, as above.

4.2 Unforgeability

Lemma 11. *With respect to the language L defined in* (5)*, the* ShowCredential *protocol is black-box extractable in the sense of Definition 17.*

Proof. By Definition 17, 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 $\bar{K}, \bar{S}, (\bar{S}_i)_{i=0,\ldots,n}, \tilde{T}$ 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 $\bar{T} = T^{\alpha} = C^{\alpha z} = (\bar{K}\bar{S}^{\kappa}\bar{S}_0^{k_0}\prod_{i=0}^{n}\bar{S}_i^{k_i})^z$,
– it returns the credential $(k_0,\ldots,k_n),(\kappa,\bar{K},\bar{S},\bar{S}_0,\ldots,\bar{S}_n,\bar{T})$.

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 $\epsilon(|x|) - \nu_C(|x|)$, where $\nu_C(|x|)$ 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_C is negligible by Definition 17, this proves the claim.

Theorem 12. *Our credential scheme is unforgeable.*

Proof. The unforgeability game (see Definition 2) closely resembles the game outlined in Definition 18 that defines unforgeability of signature schemes under adaptively chosen message attacks. Indeed, the proof of Theorem 9 that states that the underlying signature scheme is unforgeable under adaptively chosen message attacks also works here, with the following two differences:

– Instead of signing the attributes directly, the issuer (that is, the challenger) must sign them using the Issue protocol. When performing the Issue protocol, we let the challenger extract the private key k_0 and the number κ' from the proof of knowledge that the adversary performs. Since it knows κ'' and all the other attributes $k_1,\ldots,k_n$, the challenger can still solve equation (4) to κ_j.
– In the Challenge phase, the adversary must convince the challenger using the ShowCredential protocol, instead of sending his forgery directly to the challenger. If an adversary wins the unforgeability game with non-negligible, then with non-negligible probability it would be possible to extract a valid credential from the adversary by the lemma above.

With these modifications, the proof of Theorem 9 proves the statement.

4.3 Anonymity

In order to prove that our ShowCredential protocol is zero-knowledge with respect to the language L from equation (5), 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 tuple $(\hat{K},\hat{S},\hat{S}_0,\ldots,\hat{S}_n,\hat{C},\hat{T}) \in G_1^{n+5}$ such that $\hat{S} = \hat{K}^a$, $\hat{S}_i = \hat{K}^{a_i}$ and $\hat{T} = \hat{C}^z$. For that reason, we will henceforth assume that such a tuple is included with the public key σ of the credential scheme.[9] Since one can view this tuple as an extra credential of which the numbers $(\kappa,k_0,\ldots,k_n,)$ are not known, the underlying signature scheme will remain unforgeable (in the sense that Lemma 8 and Theorem 9 still hold), and for the same reason it will not impact the unforgeability of the credential scheme (see Theorem 12).

Theorem 13. *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 10), and extractable (Theorem 11), so by Definition 17 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.

[9] Notice that a verifier would obtain just such a tuple by executing the ShowCredential protocol once with an honest prover. Therefore, for the purpose of proving unlinkability (as in Definition 3) it is not strictly necessary to include these extra elements in the public key, as the simulator could just use the group elements that it obtains from one of the ShowCredential queries.

– 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 provers and the simulator, the elements $\bar{S}, \bar{S}_i$ and $\tilde{T}$ are determined by $\bar{K}$ and $\tilde{C}$ respectively.

Essentially, the behavior of the simulator deviates from honest users only in the following two ways:

– The simulator uses the simulator for the proof of knowledge over $\beta, \kappa, k_0, (k_i)_{i \in \mathcal{C}}$ instead of honestly proving knowledge of these numbers;
– The element $\tilde{C}$ does not satisfy equation (6).

However, by the black-box zero-knowledge properties of the proof of knowledge over β, κ, k_0, $(k_i)_{i \in \mathcal{C}}$ neither of these differences can be detected by the verifier. Thus the verifier $\mathcal{V}^*$ will behave no different than it would have done if it had interacted with an honest prover $\mathcal{P}$.

Theorem 14. *Our credential scheme is unlinkable.*

Proof. Follows from Theorems 13 and 4.

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

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 $i = 0, \ldots, n$, then it can calculate D by

$$D = \big(KR \prod_{i \in \mathcal{D}} R_i\big)^{-\alpha}$$

which takes 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 D, we have $\mathrm{pk}(i) = i + 6$). Then the number of exponentiations in G_1 that the prover has to do is

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

The prover performs no exponentiations in G_2 and G_T and performs no pairings. This is optimal, since elements from G_2 are bigger and more expensive to deal with than those from G_1 (because $G_1 \subset E(\mathbb{F}_q)$ 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 of Appendix D that is used for proving knowledge of the numbers $\beta, \kappa, k_0, (k_i)_{i \in \mathcal{C}}$ consists of four moves. If one is willing to assume the Random Oracle Model (ROM), however, then we can apply the Fiat-Shamir heuristic [FS87; BR93; BPW12] to the Schnorr Σ-protocol for DL-representations (see Figure 3 in Appendix D) 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). By manipulating the random oracle, one can show that the witness can be extracted from the prover if we can rewind it, and that the resulting proof of knowledge is zero-knowledge (although in a different sense than Definition 17 because of the non-interactive nature of the proof). This implies that if we use this proof of knowledge for $\beta, \kappa, k_0, (k_i)_{i \in \mathcal{C}}$, that the credential scheme remains unlinkable and unforgeable. Additionally, 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 $\text{pk}(i) = i$ 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 [CGH04; GK03]. It should also be possible to use other non-interactive zero-knowledge proofs (NIZK) that do not rely on the Random Oracle Model, but we have not looked into this.

6 Related work

The Idemix credential scheme [CL01; IBM12] 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 modulus means that, when implementing the prover on smart cards, it is difficult to get acceptable running times for the protocols. For example, in [VA13] 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 [Bra00; PZ13]. 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 verifier cannot extract or learn information about the undisclosed attributes). Moreover, there is no unforgeability proof for U-Prove credentials. 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 [BL13], which can also be implemented on elliptic curves, but the credentials are not unlinkable; and [HM13], which runs in RSA groups like Idemix.

The credential scheme from [CL04] is closer to the schemes presented here: it is unlinkable and uses elliptic curves and (Type 1) pairings. [CL04] does not contain 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 user has to perform in the ShowCredential protocol of this scheme to ours (assuming that the protocol from Appendix D 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 [CL04] 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 [HK14]. Although an unlinkable credential scheme is aimed at, the paper contains no unlinkability proof. Moreover, unfortunately we have found 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 [VRH15].

Table 1 compares the amount of exponentiations in our scheme to those of [CL04], 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.

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	$n + \lvert\mathcal{C}\rvert + 15$	0	0	0	yes
[CL04]	$2n + 3$	$\lvert\mathcal{C}\rvert + 8$	$n + 3$	0	yes
Idemix	0	0	0	$\lvert\mathcal{C}\rvert + 3$	yes
U-Prove	$\lvert\mathcal{C}\rvert + 1$	0	0	0	no

References

[AF07] M. Abe and S. Fehr. "Perfect NIZK with Adaptive Soundness". In: *Theory of Cryptography*. Ed. by S. P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 118–136 (cit. on p. 23).

[Bit+12] 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. Cambridge, Massachusetts: ACM, 2012, pp. 326–349 (cit. on p. 5).

[Bit+14] N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer. "The Hunting of the SNARK". In: *IACR Cryptology ePrint Archive* 2014 (2014). URL: `https://eprint.iacr.org/2014/580` (cit. on p. 5).

[BL13] F. Baldimtsi and A. Lysyanskaya. “Anonymous Credentials Light”. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS ’13. Berlin, Germany: ACM, 2013, pp. 1087–1098 (cit. on p. 15).

[BP04a] M. Bellare and A. Palacio. “The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols”. English. In: *Advances in Cryptology – CRYPTO 2004*. Ed. by M. Franklin. Vol. 3152. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 273–289 (cit. on p. 5).

[BP04b] M. Bellare and A. Palacio. “Towards Plaintext-Aware Public-Key Encryption Without Random Oracles”. English. In: *Advances in Cryptology - ASIACRYPT 2004*. Ed. by P. Lee. Vol. 3329. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 48–62 (cit. on p. 5).

[BPW12] D. Bernhard, O. Pereira, and B. Warinschi. “How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios”. In: *Advances in Cryptology – ASIACRYPT 2012*. Ed. by X. Wang and K. Sako. Vol. 7658. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 626–643 (cit. on p. 15).

[BR93] M. Bellare and P. Rogaway. “Random oracles are practical: a paradigm for designing efficient protocols”. In: *ACM Conference on Computer and Communications Security*. 1993, pp. 62–73 (cit. on p. 15).

[Bra00] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000 (cit. on pp. 1, 15, 21, 24).

[BSS05] I. F. Blake, G. Seroussi, and N. P. Smart, eds. *Advances in Elliptic Curve Cryptography*. Cambridge Books Online. Cambridge University Press, 2005 (cit. on p. 20).

[CDM00] R. Cramer, I. Damgård, and P. MacKenzie. “Efficient Zero-Knowledge Proofs of Knowledge without Intractability Assumptions”. In: *Public Key Cryptography*. Ed. by H. Imai and Y. Zheng. Vol. 1751. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 354–372 (cit. on pp. 20, 24).

[CDS94] 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. London, UK, UK: Springer-Verlag, 1994, pp. 174–187 (cit. on p. 24).

[CGH04] R. Canetti, O. Goldreich, and S. Halevi. “The Random Oracle Methodology, Revisited”. In: *J. ACM* 51.4 (2004), pp. 557–594 (cit. on p. 15).

[CL01] J. Camenisch and A. Lysyanskaya. “An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation”. In: *Advances in Cryptology - EUROCRYPT 2001*. Ed. by B. Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Springer, 2001, pp. 93–118 (cit. on pp. 1, 15).

[CL04] J. Camenisch and A. Lysyanskaya. “Signature Schemes and Anonymous Credentials from Bilinear Maps”. English. In: *Advances in Cryptology – CRYPTO 2004*. Ed. by M. Franklin. Vol. 3152. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 56–72 (cit. on p. 16).

[CP93] D. Chaum and T. P. Pedersen. “Wallet Databases with Observers”. In: *Advances in Cryptology - CRYPTO ’92*. Ed. by E. F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1993, pp. 89–105 (cit. on p. 5).

[CS97] J. Camenisch and M. Stadler. “Efficient group signature schemes for large groups”. In: *Advances in Cryptology — CRYPTO ’97*. Ed. by B. S. J. Kaliski. Vol. 1294. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, pp. 410–424 (cit. on p. 2).

[Dam10] I. Damgård. *On Σ-protocols*. CPT 2010, v.2. 2010. URL: `http://www.cs.au.dk/~ivan/Sigma.pdf` (cit. on p. 24).

[Dam91] I. Damgård. "Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks". In: *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. Ed. by J. Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 445–456 (cit. on pp. 4, 5).

[Den06] A. W. Dent. *The Hardness of the DHK Problem in the Generic Group Model*. Cryptology ePrint Archive, Report 2006/156. `https://eprint.iacr.org/2006/156`. 2006 (cit. on p. 23).

[DFH12] I. Damgård, S. Faust, and C. Hazay. "Secure Two-Party Computation with Low Communication". English. In: *Theory of Cryptography*. Ed. by R. Cramer. Vol. 7194. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 54–74 (cit. on p. 5).

[DG09] M. Di Raimondo and R. Gennaro. "New Approaches for Deniable Authentication". English. In: *Journal of Cryptology* 22.4 (2009), pp. 572–615 (cit. on p. 5).

[DGK06] M. Di Raimondo, R. Gennaro, and H. Krawczyk. "Deniable Authentication and Key Exchange". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. Alexandria, Virginia, USA: ACM, 2006, pp. 400–409 (cit. on p. 5).

[FS87] A. Fiat and A. Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology – CRYPTO' 86*. Ed. by A. M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1987, pp. 186–194 (cit. on p. 15).

[FS90] 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. Baltimore, Maryland, USA: ACM, 1990, pp. 416–426 (cit. on p. 24).

[GK03] 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), 11-14 October 2003, Cambridge, MA, USA, Proceedings*. IEEE Computer Society, 2003, pp. 102–113 (cit. on p. 15).

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks". In: *SIAM Journal on Computing* 17.2 (Apr. 1988), pp. 281–308 (cit. on p. 21).

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM Journal on Computing* 18.1 (1989), pp. 186–208 (cit. on p. 20).

[Gol00] O. Goldreich. *Foundations of Cryptography: Basic Tools*. New York, NY, USA: Cambridge University Press, 2000 (cit. on p. 20).

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. "Pairings for cryptographers". In: *Discrete Applied Mathematics* 156.16 (2008), pp. 3113–3121 (cit. on p. 20).

[Gro10] J. Groth. "Short Pairing-Based Non-interactive Zero-Knowledge Arguments". English. In: *Advances in Cryptology - ASIACRYPT 2010*. Ed. by M. Abe. Vol. 6477. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 321–340 (cit. on p. 5).

[HK14] L. Hanzlik and K. Kluczniak. "A Short Paper on How to Improve U-Prove Using Self-Blindable Certificates". In: *Financial Cryptography and Data Security*. Ed. by N. Christin and R. Safavi-Naini. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 273–282 (cit. on p. 16).

[HM13] J. Hajny and L. Malina. "Unlinkable Attribute-Based Credentials with Practical Revocation on Smart-Cards". English. In: *Smart Card Research and Advanced Applica-*

tions. Ed. by S. Mangard. Vol. 7771. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 62–76 (cit. on p. 15).

[IBM12] IBM Research Zürich Security Team. *Specification of the Identity Mixer Cryptographic Library, version 2.3.0*. Tech. rep. IBM Research, Zürich, Feb. 2012. URL: `https://tinyurl.com/idemix-spec` (cit. on pp. 1, 15).

[Kra05] H. Krawczyk. "HMQV: A High-Performance Secure Diffie-Hellman Protocol". English. In: *Advances in Cryptology – CRYPTO 2005*. Ed. by V. Shoup. Vol. 3621. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 546–566 (cit. on p. 5).

[Lin11] 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` (cit. on p. 24).

[PZ13] C. Paquin and G. Zaverucha. "U-Prove Cryptographic Specification V1.1 (Revision 3)". Released under the Open Specification Promise. Dec. 2013. URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=166969` (cit. on pp. 1, 15).

[Sch90] C. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology — EUROCRYPT '89*. Ed. by J.-J. Quisquater and J. Vandewalle. Vol. 434. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1990, pp. 688–689 (cit. on p. 24).

[Sho97] V. Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In: *Advances in Cryptology — EUROCRYPT '97*. Ed. by W. Fumy. Vol. 1233. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, pp. 256–266 (cit. on p. 22).

[VA13] P. Vullers and G. Alpár. "Efficient Selective Disclosure on Smart Cards Using Idemix". In: *Policies and Research in Identity Management*. Ed. by S. Fischer-Hübner, E. de Leeuw, and C. Mitchell. Vol. 396. IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, 2013, pp. 53–67 (cit. on p. 15).

[Ver01] E. R. Verheul. "Self-Blindable Credential Certificates from the Weil Pairing". In: *Advances in Cryptology - ASIACRYPT*. Ed. by C. Boyd. Vol. 2248. Lecture Notes in Computer Science. Springer, 2001, pp. 533–551 (cit. on p. 6).

[VRH15] E. Verheul, S. Ringers, and J.-H. Hoepman. *The self-blindable U-Prove scheme by Hanzlik and Kluczniak is forgeable*. Cryptology ePrint Archive, Report 2015/725. `https://eprint.iacr.org/2015/725`. 2015 (cit. on p. 16).

[WS07] J. Wu and D. R. Stinson. "An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption". In: *IACR Cryptology ePrint Archive* 2007 (2007), p. 479. URL: `https://eprint.iacr.org/2007/479` (cit. on p. 5).

[WS08] J. Wu and D. Stinson. *On The Security of The ElGamal Encryption Scheme and Damgård's Variant*. Cryptology ePrint Archive, Report 2008/200. 2008. URL: `https://eprint.iacr.org/2008/200` (cit. on p. 5).

[YZ10] A. Yao and Y. Zhao. "Deniable Internet Key Exchange". English. In: *Applied Cryptography and Network Security*. Ed. by J. Zhou and M. Yung. Vol. 6123. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 329–348 (cit. on p. 5).

A Definitions

Definition 15. 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 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*: Denoting the generators of G_1 and G_2 with $P \in G_1$, $Q \in G_2$ respectively, 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 only 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. For more information about bilinear group pairs and pairings we refer to [GPS08]; see also, for example, Chapters I and X from [BSS05].

Definition 16 (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 polynomial-time algorithm can, given a generator P and an element $A = P^a$, output a with probability that is non-negligible in ℓ.

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 17 (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 ([GMR89], see also [CDM00; Gol00] 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

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

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

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

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:

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

That is, if using a witness w the prover can get any verifier 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 $\text{view}_{\mathcal{V}^*}(\mathcal{S}(x) \overset{\blacksquare}{\to} \mathcal{V}^*(x, a))$, violating equation (7).

Definition 18. 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 [GMR88]. 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 the adversary $\mathcal{A}$.

Queries The adversary requests signatures on messages $m_1, \ldots, m_q \in \mathcal{M}$ 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 $1 < i < q$.

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 DL-representations

Definition 19. Let G be a cyclic group of prime order p and let $X_1, \ldots, X_n \in G$ all be distinct. If for some $K \in G$ the numbers $(k_1, \ldots, k_n) \in \mathbb{Z}_p^n$ are such that $K = \prod_{i=1}^n X_i^{k_i}$, then $(k_1, \ldots, k_n)$ is called a *DL-representation* of K with respect to $(X_1, \ldots, X_n)$.

Proposition 20. *Let $(X_1, \ldots, X_n)$ be randomly distributed. No probabilistic polynomial-time algorithm can, on input $(X_1, \ldots, X_n)$, generate a non-trivial DL-representation of $1 \in G$ with respect to $(X_1, \ldots, X_n)$.*

Proof. (From [Bra00, p. 60]). Suppose that such an algorithm $\mathcal{A}$ does exist. We construct an algorithm $\mathcal{B}$ that, on input $K, L \in G$ computes $\log_K L \bmod p$ using $\mathcal{A}$ as follows:

1. $\mathcal{B}$ generates $2n$ random numbers $r_1, \ldots, r_n, s_1, \ldots, s_n \in \mathbb{Z}_p$, and sets, for each i,

$$X_i = K^{r_i} L^{s_i}.$$

2. $\mathcal{B}$ then executes $(k_1, \ldots, k_n) \leftarrow \mathcal{A}(X_1, \ldots, X_n)$, and checks whether the output of $\mathcal{A}$ is indeed a non-trivial DL-representation of 1. If it is not then $\mathcal{B}$ halts.
3. If $\sum_{i=1}^n s_i k_i = 0 \bmod p$ then $\mathcal{B}$ halts.

4. $\mathcal{B}$ outputs

$$-\left(\sum_{i=1}^{n} r_i k_i\right)\left(\sum_{i=1}^{n} s_i k_i\right)^{-1} \bmod p.$$

Since

$$1 = \prod_{i=1}^{n} X_i^{k_i} = K^{\sum_{i=1}^{n} r_i k_i} L^{\sum_{i=1}^{n} s_i k_i} = K^{\sum_{i=1}^{n} r_i k_i + (\log_K L) \sum_{i=1}^{n} s_i k_i}$$

it follows that $\sum_{i=1}^{n} r_i k_i + (\log_K L) \sum_{i=1}^{n} s_i k_i = 0 \bmod p$, so that the output of $\mathcal{B}$ is indeed $\log_K L$.

It remains to show that the chance of success of $\mathcal{B}$ – that is, the chance that $\mathcal{B}$ does not terminate in step 3 – is not negligible. Let ϵ be the chance that $\mathcal{A}$ succeeds. The output $(k_1, \ldots, k_n)$ of $\mathcal{A}$ cannot depend on the numbers s_i, because they are hidden from $\mathcal{A}$ because of the randomness of the numbers r_i. For any tuple $(s_1, \ldots, s_{n-1})$ there exists exactly one number s_n which is such that $\sum_{i=1}^{n} s_i k_i = 0 \bmod p$, so that there are p^{n-1} "bad" tuples among the total of p^n tuples. Since $\mathcal{B}$ chooses the numbers s_i randomly, the change that it chooses a "bad" tuple is therefore $p^{n-1}/p^n = 1/p$, so that there is a chance $1 - 1/p$ that it chooses a "good" tuple.

Since step 4 takes place only if step 3 is successful, and step 3 takes place only if step 2 is successful, the overall success probability of $\mathcal{B}$ is $\epsilon(1 - 1/p)$ which is not negligible if ϵ is not negligible.

This proposition implies that if a polynomial-time algorithm finds such a DL-representation of 1, then the representation is trivial.

In the proposition above, all of the base points X_i are randomly distributed. In Lemma 8, where we use this inability to produce non-trivial DL-representations of 1, this is not the case: there the DL-representation is of the form

$$1 = \prod_{j=1}^{m} K_j^{k_j} S_j^{\ell_j}$$

where $S_j = K_j^a$ for each j, and this can be detected using the bilinear pairing and $A = aQ \in G_2$. The proof above can, however, easily be modified to take this into account.

C The XKEA assumption and the generic group model

The difficulty of solving a computational problem depends on how it is represented. Consider for example the DDH problem. In $\mathbb{Z}_p$ this is easy, but there exist elliptic curves over $\mathbb{Z}_p$ for which DDH is assumed to be hard, even though the two groups are isomorphic (the isomorphism being $a \mapsto P^a$ for any element P). The generic group model is a way of studying algorithms that cannot exploit the representation of group elements, but only use the group structure.

In the context of the KEA assumption, this notion is usually formalized using techniques proposed by Shoup [Sho97] (where it is also proven that the discrete logarithm problem is hard in the generic group model). In this model, algorithms are not provided direct access to group elements, but only to the images of group elements of a random bijection $\sigma\colon \mathbb{Z}_p \to \mathbb{Z}_p$, together with oracle access to the following two functions:

$$\mathsf{add}(\sigma(x), \sigma(y)) = \sigma(x + y),$$
$$\mathsf{inv}(\sigma(x)) = \sigma(-x).$$

This also provides the ability to take random elements to the algorithm, by querying the oracle on $\sigma(x')$ for any previously unseen x'. If there is a bilinear pairing, this can also be modeled using such an oracle function. It is clear that in this situation, the attacker can gain no advantage in solving a computational problem from the representation $\sigma(x)$ of an element x. In the case of the KEA assumption, if $(L, L^a) \leftarrow \mathcal{A}(K, K^a)$, one can show that with overwhelming probability the number c such that $L = K^a$ can be calculated by observing the oracle calls made by $\mathcal{A}$.

Let n denote the amount of pairs $(K_i, S_i = K_i^a)$ that our algorithm receives. The first two proofs along these lines seem to be from [Den06], where it is proved for $n = 1$, and [AF07], where it is proved for $n = 1, 2$ in the presence of a symmetric (i.e. Type 1) pairing. The latter proof can easily be modified into one that allows

- a Type 3 pairing instead of Type 1,
- $n > 2$ pairs of elements K_1, K_i^a,
- elements from G_2 as auxiliary input,[10]

resulting in the version from Definition 6.

Let us now turn to Algorithm $\mathcal{B}$ from the proof of Lemma 8, and explain why the two applications of the XKEA assumption to $\mathcal{B}$ are justified. $\mathcal{B}$ receives as parameters

$$(\kappa_j, K_j, S_j, C_j, T_j)_{j=1,\ldots,m}$$

Recall that the discrete log of S_i with respect to K_i is always a. Suppose now that the XKEA assumption relative to a does *not* apply to $\mathcal{B}$: i.e., there does not exist a polynomial-time probabilistic extractor that, if $\mathcal{B}$ outputs valid K, S, C, T, outputs $c_1, \ldots, c_n$ such that $K = K_1^{c_1} \cdots K_m^{c_m}$. We then define a new algorithm $\mathcal{C}$ as below.

Algorithm 2 Algorithm $\mathcal{C}$.

```
1: function C((K_j, S_j)_{j=1,...,m})
2:     z ∈_R Z_p^*
3:     for all j ∈ {1, ..., m} do
4:         κ_i ∈_R Z_p^*
5:         C_i ← K_i S_i^{κ_i}
6:         T_i ← C_i^z
7:     end for
8:     return B((κ_j, K_j, S_j, C_j, T_j)_{j=1,...,m})
9: end function
```

It is clear that in that case, the XKEA assumption also cannot hold for Algorithm $\mathcal{C}$, directly contradicting the XKEA assumption itself. Therefore we conclude that there must be an extractor for algorithm $\mathcal{B}$.

The second application of the XKEA assumption to Algorithm $\mathcal{B}$ relative to the secret key z, can be justified in the same way. Algorithm $\mathcal{C}$ now receives C_i, T_i for $i = 1, \ldots, m$, generates

[10] Some papers allow the algorithm to additionally receive *arbitrary* auxiliary input $\xi \in \{0,1\}^*$. We are more careful than that, because the statement in this form could impossibly hold for any group in which the discrete logarithm problem is hard: if $\xi = a$, then from the program that takes a random $C \in G_1$ and returns C, C^a, no coefficient can be extracted (besides a).

$a \in_R \mathbb{Z}_p^*$ and does the following in its for each-loop:

$$\kappa_i \in_R \mathbb{Z}_p^*$$
$$K_i \leftarrow C_i^{1/(1+a\kappa_i)}$$
$$S_i \leftarrow K_i^a.$$

Then $C_i = K_i S_i^{\kappa_i}$ as required, so that it can invoke algorithm $\mathcal{B}$ as above.

D Proofs of knowledge of discrete logarithms and DL-representations

In the protocols presented in this paper, proofs of knowledge of discrete logarithms and DL-representations occur. The Schnorr Σ-protocol [Sch90] and its extension to DL-representations (see Figure 3 and [Bra00]) 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 17).

Fortunately, there are extensions of the Schnorr Σ-protocols that are fully zero-knowledge; we present an optimization of the one from [CDM00] for DL-representations here. When $n = 1$ is taken, it reduces to a proof of knowledge for discrete logarithms.[11]

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 $(k_1, \ldots, k_n)$ of $K = P_1^{k_1} \cdots P_n^{k_n}$ with respect to $P_1, \ldots, P_n \in G$, 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 [CDS94] (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 $(k_1, \ldots, k_n)$ 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 [FS90].

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 $(k_1, \ldots, k_n)$. 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.

[11] Other protocols satisfying Definition 17 that could be used are, for example, the one sketched in [Dam10, p. 16] which also has 4 moves, or the protocol for discrete logs from [Lin11, p. 31], which is easily extended to a protocol for DL-representation but has 5 moves.

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 $K = \prod_{i=1}^{n} P_i^{k_i}$		
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 $K^c W \stackrel{?}{=} \prod_{i=1}^{n} P_i^{s_i}$

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

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 $P_1^{z_1} K^{z_2} \stackrel{?}{=} \tilde{W} W^e$		
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 $c \stackrel{?}{=} c_1 + c_2$
		verify $P_1^{z_1'} K^{z_2'} \stackrel{?}{=} W' W^e$
		verify $P_1^{\bar{z}_1} \cdots P_n^{\bar{z}_n} \stackrel{?}{=} \bar{W} K^{c_2}$

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 $(k_1, \ldots, k_n)$ of K (in which case we are done), or numbers c', z' such that (W, c', z') is accepting for 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 DL-representation of K with respect to $P_1, \ldots, P_n$).

E 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 $j \in_R \{1, \ldots, m\}$, 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 challenger 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.