Improved Algebraic MACs and Practical Keyed-Verification Anonymous Credentials

Amira Barki[1,2], Solenn Brunet[1,3](✉), Nicolas Desmoulins[1], and Jacques Traoré[1]

[1] Orange Labs, Caen, France
solenn.brunet@orange.com
[2] Sorbonne Universités, Université de Technologie de Compiègne (UTC), CNRS, UMR 7253 Heudiasyc, Compiègne, France
[3] Université de Rennes 1, Rennes, France

Abstract. Until quite recently, anonymous credentials systems were based on public key primitives. A new approach, that relies on algebraic Message Authentication Codes (MACs) in prime-order groups, has recently been introduced by Chase *et al.* at CCS 2014. They proposed two anonymous credentials systems referred to as "*Keyed-Verification Anonymous Credentials (KVAC)*" as they require the verifier to know the issuer secret key. Unfortunately, both systems presentation proof, for n unrevealed attributes, is of complexity $O(n)$ in the number of group elements. In this paper, we propose a new KVAC system that provides multi-show unlinkability of credentials and is of complexity $O(1)$ in the number of group elements while being almost as efficient as Microsoft's U-Prove anonymous credentials system (which does not ensure multi-show unlinkability) and many times faster than IBM's Idemix. Our credentials are constructed based on a new algebraic MAC scheme which is of independent interest. Through slight modifications on the verifier side, our KVAC system, which is proven secure in the random oracle model, can be easily turned into a public-key credentials system. By implementing it on a standard NFC SIM card, we show its efficiency and suitability for real-world use cases and constrained devices. In particular, a credential presentation, with 3 attributes, can be performed in only 88 ms.

Keywords: MAC · Anonymous credentials · Attributes · Multi-show unlinkability · Java Card SIM card

1 Introduction

Introduced by Chaum [16], anonymous credentials systems allow users to obtain a credential from an issuer and then, later, prove possession of this credential, in an unlinkable way, without revealing any additional information. This primitive has attracted a lot of interest as it complies with data minimization principles that consist in preventing the disclosure of irrelevant and unnecessary information. Typically, an anonymous credentials system is expected to enable users to reveal a subset of the attributes associated to their credentials while keeping the remaining ones hidden. For instance, a service provider only needs to know that a user is legitimate (*i.e.* he is authorized to access the service) without yet being able to collect personal information such as address, date of birth, etc.

Potential applications of anonymous credentials systems are numerous, including e-cash [21], public transport and electronic toll (for authentication purposes). In such applications, the system efficiency is an important requirement especially as it is usually deployed on constrained environments like smart cards.

Furthermore, it is desirable that an anonymous credentials system provides multi-show unlinkability. That is, one can prove possession of the same credential several times in an unlinkable manner. However, when it is intended for eCash applications, credentials should be one-show to prevent double spending of coins.

Related Work. One of the most prevalent anonymous credentials systems is Microsoft's U-Prove [23,24] which is based on a blind signature scheme due to Brands [6]. It is quite efficient, as it works in prime-order groups, and supports the selective disclosure of attributes. Nevertheless, U-Prove does not provide multi-show unlinkability unless the user uses a different credential at each proof of possession. Besides, to date, its security has not been formally proven.

A slightly less efficient anonymous attribute-based credentials system has been proposed by Baldimsti and Lysyanskaya [3]. Their proposal, which relies on an extension of Abe's blind signature scheme [1], is proven secure in the Random Oracle Model (ROM) under the DDH assumption. Recently, Fuchsbauer *et al.* [19] introduced another anonymous credentials system that is proven secure in the standard model. However, similarly to U-Prove, both systems are one-show (*i.e.* credential presentations are linkable if a credential is used more than once).

IBM's Identity Mixer, commonly known as Idemix [22], is built on Camenisch-Lysyanskaya (CL) signature scheme [10,11]. Unlike previously reviewed credentials systems, Idemix credentials provide multi-show unlinkability but at the cost of a less efficient proof of possession. Indeed, the used CL signatures are based on the Strong RSA assumption [4]. This implies large RSA parameters which make Idemix unsuitable for constrained devices. Despite this, Vullers and Alpár focused in [27] on the implementation of Idemix on MULTOS smart cards. Using a 1024-bit modulus, their implementation enables the presentation of a credential with three attributes, one of which is undisclosed, in 1 s. Moreover, de la Piedra *et al.* [25] addressed smart cards limited Random Access Memory (RAM) issues by proposing a RAM-efficient implementation of Idemix. Thereby, smart cards can support Idemix credentials with more than 5 attributes. Unfortunately, even with these implementation improvements, timing results far exceed the time constraints of some use cases, which limits the use of Idemix in practice.

Camenisch and Lysyanskaya introduced in [12] an efficient signature scheme defined in bilinear groups and used it to construct an anonymous credentials system. Shortly afterwards, Akagi *et al.* [2] provided a more effective Boneh Boyen-based anonymous credentials system. Recently, Camenisch *et al.* [9] proposed a Universally Composable (UC) secure anonymous credentials system that provides multi-show unlinkability and whose presentation proof is of constant size. Nevertheless, these three proposals require the prover to compute pairings and/or perform computations in $\mathbb{G}_2$. Thus, they cannot be implemented on SIM cards as the latter cannot handle such heavy computations.

Recently, Chase *et al.* [15] have opted for the use of symmetric key primitives, instead of digital signatures, so as to achieve better performances. More precisely, they used algebraic Message Authentication Codes (MACs), that relies on group operations rather than block ciphers or hash functions, as the main building block of their credentials system. Their two proposals, denoted $\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathsf{MAC}_{\mathsf{DDH}}$, assume that the issuer of the credential and the verifier share a secret key. In such a setting, the anonymous credentials system is referred to as *Keyed-Verification Anonymous Credentials* (KVAC). Unfortunately, their presentation proofs, for n unrevealed attributes, are of complexity $O(n)$ in the number of group elements. Moreover, when credential blind issuance is required, their KVAC systems do not provide perfect anonymity as they rely on ElGamal encryption to hide attributes.

As pointed out in [15], one can switch between the use of public-key and keyed-verification anonymous credentials which are more efficient. For that, whenever interacting with a new entity, the user proves the possession of a publicly verifiable credential (such as a driving license anonymous credential issued by a government on a set of attributes) and gets back a keyed-verification credential on the same attributes without disclosing them. Thus, during subsequent interactions with that entity, the user will use the keyed-verification credential for better efficiency.

Contributions. In this paper, we aim to design an anonymous credentials system that provides multi-show unlinkability while being both efficient and suitable for resource constrained environments like SIM cards (that cannot handle pairing computations). To this end, following Chase *et al.* approach [15], we first build a new algebraic MAC scheme that relies on a pairing-free variant of the Boneh Boyen signature scheme. We prove the security of our proposal, which is of independent interest, under the $q-\mathsf{SDH}$ assumption. Then, we use it to construct a practical Keyed-Verification Anonymous Credentials (KVAC) system whose presentation proof is of complexity $O(1)$ in the number of group elements and linear in the number of scalars. Our KVAC system is proven secure in the ROM under classical assumptions. Furthermore, it can be easily turned into an efficient publicly verifiable anonymous credentials system through the use of pairings *solely* on the verifier side. To show its efficiency and suitability for constrained environment, we implemented our system on a standard NFC SIM card. The proof of possession of a credential on three attributes, with one unrevealed, takes just 88 ms. This confirms its suitability for real world applications.

Organization. The paper is structured as follows. Section 2 introduces our main notation and necessary building blocks. Then, Sect. 3 presents a novel algebraic MAC scheme based on a pairing-free variant of the Boneh Boyen signature scheme. Next, Sect. 4 describes our keyed-verification anonymous credentials system as well as the way it can be turned into a traditional public-key anonymous credentials system. Finally, Sect. 5 provides efficiency and complexity evaluations as well as implementation benchmarks of our KVAC system.

2 Preliminaries

2.1 Classical Tools

Notation. To state that x is chosen uniformly at random from the set X, we use one of the two following notations $x \overset{R}{\longleftarrow} X$ or $x \in_R X$. In addition, $\vec{m}$ and $\{g_i\}_{i=1}^{l}$ respectively denote the vector $(m_1, \ldots, m_n)$ and the set $\{g_1, g_2, \ldots, g_l\}$.

Zero-Knowledge Proof of Knowledge. Zero-Knowledge Proofs of Knowledge (ZKPKs) allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that he knows some secrets verifying a given statement without revealing anything else about them. Following the usual notation introduced by Camenisch and Stadler [13], they are denoted by $\pi = \text{PoK}\{\alpha, \beta : \textit{statements about}\, \alpha, \beta\}$ where Greek letters correspond to the knowledge of $\mathcal{P}$.

A ZKPK should satisfy three properties, namely (1) *completeness* (*i.e.* a valid prover should be able to convince an honest verifier with overwhelming probability), (2) *soundness* (*i.e.* a malicious prover should be rejected with overwhelming probability), (3) *zero-knowledge* (*i.e.* the proof reveals no information about the secret(s)).

In addition to classical ZKPKs (such as a proof of knowledge of a discrete logarithm [26], a proof of knowledge of a representation [8], or a proof of equality of discrete logarithms [17]), our KVAC system relies on a ZKPK that a committed value is non-zero. Such a proof has been introduced by Brands [7].

Indeed, a prover $\mathcal{P}$ may sometimes have to convince the verifier $\mathcal{V}$ that the value x committed in $C = g^x h^w$ is non-zero, where g and h are two random generators (*i.e.* the discrete logarithm of g in the base h is unknown). To do so, $\mathcal{P}$ has to prove the knowledge of the representation of g in the bases C and h. That is, $\mathcal{P}$ has to build a ZKPK π defined as $\pi = \{\alpha, \beta, \gamma, \delta : C = g^\alpha h^\beta \wedge g = C^\gamma h^\delta\}$.

Computational Hardness Assumptions. The security of our MAC scheme and KVAC system relies on a set of computational hardness assumptions. In what follows, $\mathbb{G}$ denotes a cyclic group of prime order p.

Discrete Logarithm (DL) *Assumption.* The Discrete Logarithm assumption states that, given a generator $g \in_R \mathbb{G}$ and an element $y \in_R \mathbb{G}$, it is hard to find the integer $x \in \mathbb{Z}_p$ such that $y = g^x$.

Decisional Diffie-Hellman (DDH) *Assumption.* The Decisional Diffie-Hellman assumption states that, given a generator $g \in_R \mathbb{G}$, two elements $g^a, g^b \in_R \mathbb{G}$ and a candidate $X \in \mathbb{G}$, it is hard to decide whether $X = g^{ab}$ or not. This is equivalent to decide, given g, h, g^a, g^b, whether $a = b$ or not.

q-Strong Diffie-Hellman ($q - \mathsf{SDH}$) *Assumption.* The q-Strong Diffie-Hellman assumption holds in $\mathbb{G}$ if, given a generator $g \in_R \mathbb{G}$ and $(g^y, g^{y^2}, \ldots, g^{y^q}) \in \mathbb{G}^q$ as input, it is hard to output a pair $(x, g^{\frac{1}{y+x}}) \in \mathbb{Z}_p^* \times \mathbb{G}$.

This assumption is believed to be hard even in gap-DDH groups, *i.e.* groups in which there is an efficient test to determine, with probability 1, on input (g, h, g^x, h^y) if $x = y \bmod p$ or not. Moreover, it has been proven in [20] that the hardness of the $q - \mathsf{SDH}$ assumption in gap-DDH groups implies the hardness of the *gap* $q - \mathsf{SDH} - \mathsf{III}$ assumption defined as follows[1].

Gap q-Strong Diffie-Hellman-III (*gap* $q - \mathsf{SDH} - \mathsf{III}$) *Assumption.* The q-Strong Diffie-Hellman-III assumption states that, given $(g, h, g^y) \in \mathbb{G}^3$ and q distinct triples $(x_i, m_i, (g^{m_i}h)^{\frac{1}{y+x_i}}) \in \mathbb{Z}_p^2 \times \mathbb{G}$ and having access to a DDH oracle (which indicates whether a given quadruple $(g, h, g^x, h^y) \in \mathbb{G}^4$ is a DH quadruple or not), it is hard to output a *new* triple $(x, m, (g^m h)^{\frac{1}{y+x}})$ where $(x, m) \in \mathbb{Z}_p^2$.

2.2 Message Authentication Codes (MACs)

A Message Authentication Code (MAC) is an authentication tag computed using a secret key that is shared between the issuer and the verifier. More formally, a MAC scheme consists of the following four algorithms:

`Setup`(1^k) creates the public parameters pp, given a security parameter k.

`KeyGen`(pp) generates the secret key sk that is shared between the issuer and the verifier.

`MAC`(pp, sk, m) takes as input a message m and a secret key sk. It outputs a MAC, also known as a tag and denoted by τ, on the message m.

`Verify`(pp, sk, m, τ) is a deterministic algorithm which outputs either 1 or 0 depending on the validity of the MAC τ with respect to the message m and the secret key sk.

UF-CMVA Security. Usually, a probabilistic MAC scheme is considered secure if it is *unforgeable under chosen message and verification attack* (UF-CMVA). In other words, the adversary $\mathcal{A}$ can query two oracles: $\mathcal{O}$`MAC` and $\mathcal{O}$`Verify`. $\mathcal{O}$`MAC` provides him with a valid MAC on any message of his choice whereas $\mathcal{O}$`Verify` enables him to check the validity of any (message, MAC) pair. Such an adversary should not be able to compute a pair (m', τ') where τ' is a valid MAC on the message m' that has not already been queried to the $\mathcal{O}$`MAC` oracle. A yet stronger security notion for probabilistic MACs, denoted sUF-CMVA, exists. In such a variant, the adversary wins even if m' has already been queried to the $\mathcal{O}$`MAC` oracle, provided that the oracle did not output the pair (m', τ'). More formally, Fig. 1 details the sUF-CMVA experiment $\mathtt{Exp}_{\mathcal{A}}^{\text{sUF-CMVA}}(1^k)$ between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The adversary's success probability, denoted by $\mathtt{Adv}_{\mathcal{A}}^{\text{sUF-CMVA}}(1^k)$, is defined as $\Pr[\mathtt{Exp}_{\mathcal{A}}^{\text{sUF-CMVA}}(1^k) = 1]$.

[1] For this reason, we will sometimes simply refer to the *gap* $q - \mathsf{SDH} - \mathsf{III}$ assumption as the $q - \mathsf{SDH}$ assumption.

```
$\mathtt{Exp}_{\mathcal{A}}^{\text{sUF-CMVA}}(1^k)$

1. $pp \leftarrow \mathtt{Setup}(1^k)$
2. $sk \leftarrow \mathtt{KeyGen}(pp)$
3. $(m', \tau') \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{MAC}, \mathcal{O}\mathtt{Verify}}(pp)$
4. If $(m', \tau')$ was obtained following a call to the $\mathcal{O}\mathtt{MAC}$ oracle, then
   return 0.
5. Return $\mathtt{Verify}(pp, sk, m', \tau')$
```

Fig. 1. sUF-CMVA security

3 An Algebraic MAC Scheme Based on Boneh-Boyen Signatures

Based on a *pairing-free* variant [14] of the Boneh-Boyen signature scheme [5], we design a new algebraic MAC scheme. In this section, we detail our construction which can be applied to both a single message as well as a block of messages.

3.1 $\mathsf{MAC}_{\mathsf{BB}}$

Our algebraic MAC scheme for a single message m, referred to as $\mathsf{MAC}_{\mathsf{BB}}$, is defined as follows:

$\mathtt{Setup}(1^k)$ creates the system public parameters $pp = (\mathbb{G}, p, h, g_0, g_1, g)$ where $\mathbb{G}$ is a cyclic group of prime order p, a k-bit prime, and h, g_0, g_1, g are four random generators of $\mathbb{G}$.

$\mathtt{KeyGen}(pp)$ selects a random value $y \in_R \mathbb{Z}_p$ as the issuer's private key and *optionally* computes the corresponding public key $Y = g_0^y$.

$\mathtt{MAC}(m, y)$ picks two random values $r, s \in_R \mathbb{Z}_p$ and computes $A = (g_1^m g^s h)^{\frac{1}{y+r}}$. The MAC on the message m consists of the triple (A, r, s).

$\mathtt{Verify}(m, A, r, s, y)$ checks the validity of the MAC (A, r, s) with respect to the message m. The MAC is valid only if $(g_1^m g^s h)^{\frac{1}{y+r}} = A$.

Theorem 1. *Our* $\mathsf{MAC}_{\mathsf{BB}}$ *scheme is* sUF-CMVA secure *under the gap* $q - \mathsf{SDH} - \mathsf{III}$ *assumption*[2].

3.2 $\mathsf{MAC}_{\mathsf{BB}}^n$

Our algebraic MAC scheme can be generalized to support a block of n messages $(m_1, \ldots, m_n)$. This extension is referred to as $\mathsf{MAC}_{\mathsf{BB}}^n$ and works as follows:

$\mathtt{Setup}(1^{\mathtt{k}})$ creates the system public parameters $pp = (\mathbb{G}, p, g_1, g_2, \ldots, g_n, h, g_0, g)$ where $\mathbb{G}$ is a cyclic group of prime order p, a k-bit prime, and $h, g, g_0, g_1, \ldots, g_n$ are random generators of $\mathbb{G}$.

[2] The proof is detailed in Appendix A.1.

`KeyGen`(pp) selects a random value $y \in_R \mathbb{Z}_p$ as the issuer's private key and *optionally* computes the corresponding public key $Y = g_0^y$.

`MAC`($\overrightarrow{m}, y$) takes as input a block of n messages $\overrightarrow{m} = (m_1, \ldots, m_n)$ and computes $A = (g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}}$ where $r, s \in_R \mathbb{Z}_p$. The MAC on $\overrightarrow{m}$ consists of the triple (A, r, s).

`Verify`($\overrightarrow{m}, A, r, s, y$) checks the validity of the MAC with respect to the block of messages $\overrightarrow{m}$. The MAC is valid only if $(g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}} = A$.

Theorem 2. *Our* $\mathsf{MAC}_{\mathsf{BB}}^n$ *scheme is* sUF-CMVA secure *under the assumption that* $\mathsf{MAC}_{\mathsf{BB}}$ *is sUF-CMVA*[3]*.*

One particular feature of our algebraic MAC scheme is that anyone can verify the validity of a given MAC by himself (*i.e.* without neither knowing the private key y nor querying the `Verify` algorithm). Indeed, a MAC on $\overrightarrow{m} = (m_1, \ldots, m_n)$ consists of the triple (A, r, s) such that $A = (g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h)^{\frac{1}{y+r}}$. This implies that $A^{y+r} = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h$ and hence, $B = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^s h \cdot A^{-r} = A^y$. Therefore, if the issuer of the MAC also provides a ZKPK defined as

$$\pi = \mathrm{PoK}\{\gamma : B = A^\gamma \wedge Y = g_0^\gamma\},$$

then its receiver will be convinced that the MAC is valid.

Furthermore, unlike both algebraic MAC schemes due to Chase *et al.* [15], the issuer does not have to hold as many private keys as messages but rather a sole private key regardless of the number of messages.

4 A Keyed-Verification Anonymous Credentials System Based on $\mathsf{MAC}_{\mathsf{BB}}^n$

In this section, we first define Keyed-Verification Anonymous Credentials (KVAC) systems as well as their requirements. Next, we detail our new KVAC system that is built upon our $\mathsf{MAC}_{\mathsf{BB}}^n$ scheme.

4.1 Overview on KVAC Systems

A keyed-verification anonymous credentials system is defined through the following algorithms which involve three entities: a user $\mathcal{U}$, an issuer $\mathcal{I}$ and a verifier $\mathcal{V}$.

`Setup`(1^k) creates the system public parameters pp, given a security parameter k.

`CredKeyGen`(pp) generates the issuer's private key sk, which is shared with $\mathcal{V}$, and computes the corresponding public key pk.

`BlindIssue`($\mathcal{U}(\overrightarrow{m}, s), \mathcal{I}(sk)$) is an interactive protocol between a user $\mathcal{U}$ who wants to get an anonymous credential on a set of attributes $\overrightarrow{m} = (m_1, \ldots, m_n)$ and a secret value s, without revealing them, and the issuer $\mathcal{I}$ who holds the private key sk. If the protocol does not abort, the user gets a credential σ.

[3] The proof is detailed in Appendix A.2.

`Show`$(\mathcal{U}(s, \sigma, \overrightarrow{m}, \phi), \mathcal{V}(sk, \phi))$ is an interactive protocol between $\mathcal{U}$, who wants to prove that he holds a valid credential on attributes $\overrightarrow{m}$ satisfying a given set of statements ϕ, and $\mathcal{V}$, holding the private key sk, whose goal is to check that it is actually true.

Security Requirements. In addition to the usual *correctness* property, a KVAC system should satisfy four security properties, namely *unforgeability*, *anonymity*, *blind issuance* and *key-parameter consistency*. Roughly speaking, they are defined as follows (formal definitions are provided in [15]):

– *Unforgeability:* it should be infeasible for an adversary to generate a valid ZKPK that convinces a verifier that he holds a credential satisfying a given statement, or a set of statements, when it is not actually true;
– *Anonymity:* the presentation proof produced during the protocol `Show` reveals nothing else aside from the statement ϕ being proven;
– *Blind issuance:* `BlindIssue` is a secure two-party protocol for generating credentials on the user's attributes;
– *Key-parameter consistency*: an adversary should not be able to find two secret keys that correspond to the same issuer's public key.

4.2 Our Keyed-Verification Anonymous Credentials System

Based on the designed $\mathsf{MAC}_{\mathsf{BB}}^{n}$ scheme, we construct a KVAC system involving a user $\mathcal{U}$, an issuer $\mathcal{I}$ and a verifier $\mathcal{V}$. Our KVAC system consists of the following four phases. The two main phases (`BlindIssue` and `Show`) are depicted in Fig. 2.

Setup. Generate the public parameters $pp = (\mathbb{G}, p, g_1, g_2, \ldots, g_n, g, h, g_0, f)$ where $\mathbb{G}$ is a cyclic group of prime order p, a k-bit prime, and $(h, g, g_0, \{g_i\}_{i=1}^{n}, f)$ are random generators of $\mathbb{G}$ where DDH is hard. For $i \in \{1, \ldots, n\}$, g_i is associated with a specific type of attributes (*e.g.* age, gender, etc.). This allows us to differentiate attributes and avoid any ambiguity. Note that, in the sequel, all computations on exponents are computed modulo p (*i.e.* mod p).

Key Generation. Choose a random value $y \in_R \mathbb{Z}_p$ as the issuer's private key and compute the corresponding public key $Y = g_0^y$. Each user $\mathcal{U}$ is also provided with a private key sk_u and the associated public key pk_u which may be used to authenticate the user during the issuance of his credentials.

Blind Issuance. To issue a credential on the attributes $(m_1, \ldots, m_n)$, the issuer and the user (who has already been authenticated) engage in the following protocol. First, the user $\mathcal{U}$ builds a commitment $C_m = g_1^{m_1} \ldots g_n^{m_n} g^s$ on his attributes, where $s \in_R \mathbb{Z}_p^*$. Then, he sends it to the issuer $\mathcal{I}$ along with a ZKPK π_1 defined as $\pi_1 = \mathrm{PoK}\{\alpha_1, \ldots, \alpha_{n+1} : C_m = g_1^{\alpha_1} g_2^{\alpha_2} \ldots g_n^{\alpha_n} g^{\alpha_{n+1}}\}$. If the proof is valid, $\mathcal{I}$ randomly picks $r, s' \in_R \mathbb{Z}_p$ and computes $A = (C_m \cdot g^{s'} \cdot h)^{\frac{1}{y+r}}$ which corresponds to a $\mathsf{MAC}_{\mathsf{BB}}^{n}$ on $(m_1, \ldots, m_n)$. He may also build a ZKPK π_2 ensuring that the credential is well-formed. Such a proof is defined as $\pi_2 = \mathrm{PoK}\{\gamma : B = A^{\gamma} \wedge Y = g_0^{\gamma}\}$

where $B = C_m \cdot g^{s'} \cdot h \cdot A^{-r} = A^y$. Then, he provides $\mathcal{U}$ with the triple (A, r, s') along with the proof π_2. Upon receiving them, $\mathcal{U}$ first verifies the validity of π_2, then computes $\tilde{C}_m = C_m\ g^{s'}h$ as well as $s_u = s + s'$, which is a secret value only known to $\mathcal{U}$. Finally, he sets his anonymous credential σ as $\sigma = (A, r, s_u, \tilde{C}_m)$.

Note that in case where $\mathcal{U}$ does not mind revealing his attributes (or a subset of them), he just sends them without using any commitment (respectively, only commits to the attributes that he does not want to reveal).

Public Input: pp, pk_u, Y		
(1) Issuance of a credential (`BlindIssue`)		
User $\mathcal{U}$		**Issuer $\mathcal{I}$**
Private Input: $sk_u \in_R \mathbb{Z}_p^*$ $\vec{m} = (m_1, \ldots, m_n)$		**Private Input:** $y \in_R \mathbb{Z}_p^*$
Choose $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$ **Compute** $C_m \leftarrow g_1^{m_1} \cdots g_n^{m_n} g^s$ **Build**		
$\pi_1 = \text{PoK}\{\alpha_1, \cdots, \alpha_{n+1} : C_m = g^{\alpha_{n+1}} \prod_{i=1}^{n} g_i^{\alpha_i}\}$	$\xrightarrow{C_m, \pi_1}$	**Check** π_1 **and Choose** $r, s' \overset{R}{\leftarrow} \mathbb{Z}_p^*$ **Compute** $A \leftarrow (C_m \cdot g^{s'} \cdot h)^{\frac{1}{y+r}}$
Check π_2 **Compute** $\tilde{C}_m \leftarrow C_m \cdot g^{s'} \cdot h$ and $s_u \leftarrow s + s'$ $\sigma \leftarrow (A, r, s_u, \tilde{C}_m)$	$\xleftarrow{A, r, s', \pi_2}$	**Build** $\pi_2 = \text{PoK}\{\gamma : Y = g_0^{\gamma} \wedge$ $A^{\gamma} = C_m \cdot g^{s'} h \cdot A^{-r}\}$
(2) Proving Knowledge of a Credential (`Show`)		
User $\mathcal{U}$		**Verifier $\mathcal{V}$**
Private Input: $(A, r, s_u, \tilde{C}_m)$, $\vec{m}$		**Private Input:** y
Choose $l, t \overset{R}{\leftarrow} \mathbb{Z}_p^*$ **Compute** $B_0 \leftarrow A^l$, $E \leftarrow C^{\frac{1}{l}} \cdot f^t$ $C \leftarrow \tilde{C}_m^l \cdot B_0^{-r}$		
Build $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \delta_2, \ldots, \delta_{n+1}, \gamma, \theta :$ $E \cdot h^{-1} = g_1^{\delta_1} g_2^{\delta_2} \ldots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} \wedge E = C^{\alpha} f^{\beta}$ $\wedge\ C = E^{\theta} f^{\gamma}\}$	$\xrightarrow{B_0, C, E, \pi_3}$	**Compute** $C' \leftarrow B_0^y$ **Check if** $C' \overset{?}{=} C$ **Check** π_3

Fig. 2. Our keyed-verification anonymous credentials system

Credential Presentation. To anonymously prove that he holds a credential on the attributes $(m_1, \ldots, m_n)$, the user engages in an interactive protocol with the verifier $\mathcal{V}$. First, he randomly selects $l, t \in_R \mathbb{Z}_p^*$ and computes $B_0 = A^l$, a randomized version of his credential. He also computes $C = \tilde{C}_m^l B_0^{-r}$ as well as $E = C^{\frac{1}{l}} f^t$.

Note that by definition, $A^{y+r} = C_m\ g^{s'}h = g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} g^{s_u} h$. Thus, we have $(A^l)^{y+r} = g_1^{lm_1} g_2^{lm_2} \ldots g_n^{lm_n} g^{ls_u} h^l$. Hence, C is simply equal to $A^{ly} = B_0^y$.

$\mathcal{U}$ also builds a ZKPK π_3 to prove that he really holds a valid credential (*i.e.* he knows the associated attributes/secrets and the value committed in E is different from zero). π_3 is defined as $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \ldots, \delta_{n+1}, \gamma, \theta : E = C^{\alpha} f^{\beta} \wedge E \cdot h^{-1} = g_1^{\delta_1} \ldots g_n^{\delta_n} g^{\delta_{n+1}} \cdot B_0^{\lambda} \cdot f^{\beta} \wedge C = E^{\theta} f^{\gamma}\}$. Once the required values have been computed, $\mathcal{U}$ provides $\mathcal{V}$ with B_0, C and E along with π_3[4].

[4] π_3 is detailed in Appendix C.

Upon their receipt, $\mathcal{V}$ first computes $C' = B_0^y$, then verifies that $C = C'$. If so, he checks that π_3 is valid. $\mathcal{V}$ is convinced that $\mathcal{U}$ really holds a valid credential on attributes $(m_1, \ldots, m_n)$ if, and only if, both checks succeed.

Theorem 3. *Our KVAC system is* unforgeable *under the assumption that* $\mathsf{MAC}_{\mathsf{BB}}^n$ *is sUF-CMVA,* perfectly anonymous *and ensures* blind issuance *as well as* key-parameter consistency *in the Random Oracle Model*[5].

4.3 From Keyed-Verification to Public Key Anonymous Credentials

In this section, we explain how to turn our KVAC system into a public key anonymous credentials system. Thereby, a user would be able to prove possession of a credential to any entity (*i.e.* even if the issuer's private key is unknown).

For that, our system should be defined in bilinear groups. Let us first recall that bilinear groups are a set of three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ satisfying the following properties:

- For all $g \in \mathbb{G}_1, \tilde{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p, e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{a.b}$;
- For $g \neq 1_{\mathbb{G}_1}$ and $\tilde{g} \neq 1_{\mathbb{G}_2}$, $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$;
- e is efficiently computable.

In such a case, the system public parameters are defined as $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p,\ e, g_1, \ldots, g_n, g, h, g_0, f, \tilde{g}_0)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of prime order p, $(h, g, g_0, \{g_i\}_{i=1}^n, f)$ are random generators of $\mathbb{G}_1$ and $\tilde{g}_0$ is a random generator of $\mathbb{G}_2$. The other phases are updated as follows.

Key Generation. The issuer publishes a second public key $W = \tilde{g}_0^y$ associated with his private key y.

Blind Issuance. This phase does not require any changes.

Credential Presentation. As the verifier $\mathcal{V}$ does not hold the private key y, some changes are required on his side. More precisely, he must compute two pairings $e(C, \tilde{g}_0)$ and $e(B_0, W)$. $\mathcal{V}$ is convinced that the user really holds a valid credential on $(m_1, \ldots, m_n)$ only if $e(C, \tilde{g}_0) = e(B_0, W)$ and π_3 is valid.

5 Efficiency Comparison and Performance Assessment

We first compare the efficiency of our KVAC system to that of the main existing anonymous credentials schemes (*i.e.* U-Prove, Idemix, Bilinear CL, $\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathsf{MAC}_{\mathsf{DDH}}$) both in terms of credential size and computational cost related to the creation of a presentation proof since it is the most time-critical phase. Next, we focus on the complexity, in the number of group elements, of KVAC systems presentation proofs. Finally, we provide timing results of the implementation of our *Credential presentation* protocol on a standard NFC SIM card.

[5] Proofs are detailed in Appendix B.

Presentation Proof Computational Cost. We compare in Table 1 the estimated cost of creating a presentation proof in terms of total number of multi-exponentiations. We use the same notation as [15] where l-exp denotes the computation of the product of l powers and $l - exp(b_1, \ldots, b_l)$ corresponds to the computation of the product of l powers with exponents of $b_1, \ldots, b_l$ bits (for Idemix). The number of multi-exponentiations depends on three parameters: n, r and c which respectively denote the number of attributes in a credential, the number of revealed attributes and the number of attributes kept secret.

Table 1 shows that our KVAC system is competitive with U-Prove (which does not provide multi-show unlinkability) and $\mathsf{MAC}_{\mathsf{GGM}}$ (which requires the verifier to know the issuer's private key and thus does not allow public verifiability). When most of the attributes are not disclosed, our proposal outperforms $\mathsf{MAC}_{\mathsf{GGM}}$.

Table 1. Comparison of credential sizes (for s *unlinkable* shows) and presentation proof generation cost (for a credential on n attributes, c of which are not disclosed). Note that all schemes use a 256-bit elliptic curve group, except Idemix which uses a 2048-bit modulus.

Schemes	Credential size (in bits)	Number of exponentiations
U-Prove [23,24]	$1024s$	$2c$ 2-exp and 1 $(n - r + 1)$-exp
Idemix [22]	5369	1 1-exp(2048), c 2-exp(256, 2046), c 2-exp(592, 2385) and 1 $(n - r + 2)$-exp (456, 3060, 592, ..., 592)
Bilinear CL [12]	$512n + 768$	$(3 + n)$ 1-exp, $2c$ 2-exp and $3 + n$ pairings
$\mathsf{MAC}_{\mathsf{GGM}}$ [15]	512	3 1-exp, $2(n - r)$ 2-exp and 1 $(n - r + 1)$-exp
$\mathsf{MAC}_{\mathsf{DDH}}$ [15]	1024	6 1-exp, $2(n - r + 1)$ 2-exp and 2 $(n - r + 1)$-exp
$\mathbf{MAC}^{n}_{\mathbf{BB}}$	**1024**	**1 1-exp, 4 2-exp and 1 $(n - r + 3)$-exp**

Complexity in the Number of Group Elements. As it only requires a multi-commitment to all undisclosed attributes, our presentation proof is of complexity $O(1)$ in the number of group elements. This makes our KVAC system more efficient than Chase *et al.* systems (*i.e.* $\mathsf{MAC}_{\mathsf{GGM}}$ and $\mathsf{MAC}_{\mathsf{DDH}}$ [15]) whose presentation proof is of complexity $O(c)$. Indeed, both of their proposals presentation proof needs c commitments (one for each unreavealed attribute).

Implementation Results. Table 2 gives timing results of the implementation of our `Show` protocol on a Javacard 2.2.2 SIM card, Global Platform 2.2 compliant, embedded in a Samsung galaxy S3 NFC smartphone. Compared to the javacard specifications, the only particularity of our card is some additional API provided by the card manufacturer enabling operations in modular and elliptic curve arithmetic. To be able to handle asymmetric cryptography on elliptic curves, the used card is equipped with a cryptoprocessor. This makes it more powerful than most cards. It is, however, worth to emphasize that such SIM cards are already widely deployed by some phone carriers to provide NFC based services.

The implementation uses a 256-bit prime "pairing friendly" Barreto-Naehrig elliptic curve. In our implementation, the protocol is split into two parts: an *off-line* part that can be run in advance by the card (during which all the values necessary for an execution of the `Show` protocol in the worst case scenario, i.e. no revealed attributes, are computed) and an *on-line* part that needs to be performed on-line as it depends on the verifier's challenge. Indeed, in our implementation, the proof π_3 is made non-interactive: the verifier sends to the prover a challenge Ch which is included in the computation of the hash value c. Timings are given for $n = 3$, $r = 2$ and $c = 1$.

Table 2. Timings in ms ((min-max) average) of the implementation of the protocol `Show`

Off-line part (card) Battery-On: (1352–1392) 1378 ms			
On-line part			
Presentation proof (card)		*Proof verification (PC)*	
Battery-On	**Battery-Off**	***y* known**	***y* unknown**
(81–86) 83 ms	(123–124) 123.4 ms	(3–14) 5 ms	(5–17) 10 ms
Total On-line part			
Battery-On		**Battery-Off**	
***y* known**	***y* unknown**	***y* known**	***y* unknown**
(84–100) 88 ms	(86–103) 93 ms	(126–137) 128 ms	(128–141) 133 ms

The *presentation proof* by the card actually refers to the total time, from the applet selection to the proof reception, including the sending of the challenge by the verifier, but excluding the proof verification. Communication between the SIM card in the smartphone and the PC (Intel Xeon CPU 3.70 GHz), acting as the Verifier, was done in NFC using a standard PC/SC reader (an Omnikey 5321). "Battery-Off" denotes a powered-off phone either by the user, or because its battery is flat. In such a situation, as stated by NFC standards, NFC-access to the SIM card is still possible, but with degraded performances. Off-line computations are assumed to be automatically launched by the smartphone (battery-On) after a presentation proof, in anticipation for the next one. It is noteworthy that all computations are entirely done by the card: the smartphone is only used to trigger the `Show` protocol and to power the card. On-line computations refer to computations of R_i values and the hash c involved in the proof π_3 (see Appendix C), and can be potentially carried out even by a battery-Off phone. On average, the On-line part of the presentation proof is very fast even when the phone is powered-off. Actually, data exchange is the most time-consuming task.

6 Conclusion

In this paper, our contribution is twofold. First, we proposed a new algebraic MAC scheme that relies on a pairing-free variant of the Boneh Boyen signature scheme. Then, based on it, we designed a keyed-verification anonymous credentials (KVAC) system whose presentation proof is efficient both in terms of presentation cost and complexity (in the number of group elements). Our KVAC system provides multi-show unlinkability and requires the issuer to hold a single private key regardless of the number of attributes. Through slight modifications (solely on the verifier side), our KVAC system can be easily turned into a quite efficient public key anonymous credentials system. Thereby, it can also be used even if the verifier does not hold the issuer's private key. Finally, implementation results confirm its efficiency and suitability for delay sensitive applications, even when implemented on a standard NFC SIM card.

A MAC Security

A.1 Security Proof of $\mathsf{MAC_{BB}}$ (Theorem 1)

Let $\mathcal{A}$ be an adversary who breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme with non-negligible probability. Using $\mathcal{A}$, we construct a reduction $\mathcal{B}$ against the $q-\mathsf{SDH}$ assumption in gap-DDH groups (which implies the *gap* $q-\mathsf{SDH}-\mathsf{III}$ assumption). $\mathcal{A}$ can ask for tags on any message of his choice and receives the corresponding tags (A_i, r_i, s_i) for $i \in \{1, \ldots, q\}$ where q denotes the number of requests to the $\mathcal{O}\mathtt{MAC}$ oracle. Eventually, $\mathcal{A}$ outputs his forgery (A, r, s) for the message m. We distinguish two types of forgeries:

- **Type-1 Forger:** an adversary that outputs a valid tag (A, r, s) on m such that $(A, r) \neq (A_i, r_i)$ for all $i \in \{1, \ldots, q\}$.
- **Type-2 Forger:** an adversary that outputs a valid tag (A, r, s) on m such that $(A, r) = (A_j, r_j)$ for some $j \in \{1, \ldots, q\}$ and $(m, s) \neq (m_j, s_j)$.

We show that, regardless of their type, both adversaries can be used to break the gap $q-\mathsf{SDH}$ assumption. However, the reduction works differently for each type of forger. Consequently, $\mathcal{B}$ initially chooses a random bit $c_{mode} \in \{1, 2\}$ which indicates its guess for the type of forgery that $\mathcal{A}$ will output.

• If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its $q-\mathsf{SDH}$ challenger, denoted by $\mathcal{C}$, the public parameters (g_0, g_1, h) and the public key $Y = g_0^y$ as well as q random, and distinct, triples (A_i, r_i, m_i) such that $A_i = (g_1^{m_i} h)^{\frac{1}{r_i+y}}$ for $i \in \{1, \ldots, q\}$. As it is against the *gap* $q-\mathsf{SDH}-\mathsf{III}$ assumption, $\mathcal{B}$ has access to a DDH oracle, denoted by $\mathcal{O}\mathtt{DDH}$, that decides whether a given quadruple (g, h, g^x, h^y) is a valid Diffie-Hellman quadruple (*i.e.* whether $x \overset{?}{=} y \mod p$) or not. $\mathcal{B}$ also randomly chooses $v \in_R \mathbb{Z}_p$ and computes $g = g_1^v$. Thereby, it can provide $\mathcal{A}$ with the public parameters (g_0, g_1, h, g, Y) and answer his requests as follows:

- $\mathcal{O}$MAC requests: given m as input, $\mathcal{B}$ first computes s_i such that $m + vs_i = m_i$. Then, it provides $\mathcal{A}$ with the triple (A_i, r_i, s_i) which is a valid MAC on m (*i.e.* $A_i^{y+r_i} = g_1^m g^{s_i} h$). The simulation of this oracle is perfect.
- $\mathcal{O}$Verify requests: given a quadruple (A, r, s, m), $\mathcal{B}$ first verifies that $A \neq 1$ and computes $B = A^{-r} g_1^m g^s h$. Then, it provides the quadruple (g_0, A, Y, B) as input to the $\mathcal{O}$DDH oracle so as to know if it is valid or not. $\mathcal{B}$ forwards the oracle's answer to $\mathcal{A}$, thus perfectly simulating $\mathcal{O}$Verify.

Eventually, after q queries to $\mathcal{O}$MAC and q_v queries to $\mathcal{O}$Verify, $\mathcal{A}$ outputs his forgery (A, r, s) on m such that it breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme. Using these values, $\mathcal{B}$ computes $\tilde{m} = m + sv$ and outputs his forgery $(A, r, \tilde{m})$ thus breaking the $q - \mathsf{SDH}$ assumption with the same advantage as $\mathcal{A}$.

• If $c_{mode} = 2$: A Type-2 adversary $\mathcal{A}$ is rather used, as a subroutine, to construct a reduction $\mathcal{B}$ against the DL problem. In such a case, $\mathcal{B}$ receives on input from its DL challenger, denoted by $\mathcal{C}$, the challenge $(g_1, H = g_1^v)$. Its goal is to find the value v. For this purpose, it first randomly chooses $(y, g_0, h) \in_R \mathbb{Z}_p \times \mathbb{G}^2$ and computes $Y = g_0^y$. $\mathcal{B}$ also sets g as $g = H$. Thereby, it can provide $\mathcal{A}$ with the public parameters (g_1, g_0, h, g, Y) and answer his requests as follows:

- $\mathcal{O}$MAC requests: as it holds y, $\mathcal{B}$ can generate a valid MAC (A, r, s) on any queried message m. To do so, it computes $A = (g_1^m g^s h)^{\frac{1}{y+r}}$ where $r, s \in \mathbb{Z}_p^*$.
- $\mathcal{O}$Verify requests: given a quadruple (A, r, s, m), $\mathcal{B}$ computes $\tilde{A} = (g_1^m g^s h)^{\frac{1}{y+r}}$. To check its validity, and answer $\mathcal{A}$'s query, $\mathcal{B}$ verifies whether $\tilde{A} \overset{?}{=} A$.

Eventually, after q queries to $\mathcal{O}$MAC and q_v queries to $\mathcal{O}$Verify, $\mathcal{A}$ outputs his forgery (A, r, s) on m such that it breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme. By assumption, (A, r) is equal to one of the (A_j, r_j) pairs output by the $\mathcal{O}$MAC oracle following $\mathcal{A}$'s request for some $j \in \{1, \ldots, q\}$. Since $(A, r) = (A_j, r_j)$, then $A^{y+r_j} = g_1^{m_j} g^{s_j} h = A^{y+r} = g_1^m g^s h$ and so $g_1^{m_j} g^{s_j} = g_1^m g^s$. We therefore necessarily have $s_j \neq s$, otherwise this would imply that $m = m_j$ (contradicting the fact that we have supposed $(m, s) \neq (m_j, s_j)$). Thereby, $g = (g_1)^{\frac{m-m_j}{s_j - s}}$. Using the values (m, m_j, s, s_j), $\mathcal{B}$ can recover v, hence breaking the DL problem. If $\mathcal{B}$ can break the DL problem, then it can break the $q - \mathsf{SDH}$ problem (by finding the discrete logarithm y of g^y in the base g).

$\mathcal{B}$ can guess which type of forgery a particular adversary $\mathcal{A}$ will output with probability 1/2. So, $\mathcal{B}$ can break the gap $q - \mathsf{SDH}$ problem with probability $\varepsilon/2$ where ε is the probability that $\mathcal{A}$ breaks the sUF-CMVA security of our $\mathsf{MAC_{BB}}$ scheme. Therefore, under the gap $q - \mathsf{SDH}$ assumption, our $\mathsf{MAC_{BB}}$ scheme is sUF-CMVA secure.

A.2 Security Proof of $\mathsf{MAC}^n_{\mathsf{BB}}$ (Theorem 2)

Let $\mathcal{A}$ be an adversary who breaks the unforgeability of our $\mathsf{MAC}^n_{\mathsf{BB}}$ with non-negligible probability. Using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$ against the unforgeability of $\mathsf{MAC_{BB}}$. $\mathcal{A}$ can ask for tags on blocks of messages $\overrightarrow{m_1} = (m_1^1, \ldots, m_n^1)$,

$\vec{m}_2 = (m_1^2, \ldots, m_n^2), \ldots, \vec{m}_q = (m_1^q, \ldots, m_n^q)$ and receives the corresponding tags (A_i, r_i, s_i) for $i \in \{1, \ldots, q\}$. Eventually, $\mathcal{A}$ outputs his forgery (A, r, s) for the block of messages $\vec{m} = (m_1, \ldots, m_n)$. We differentiate two types of forgers:

– **Type-1 Forger:** an adversary that outputs a forgery where $(A, r, s) \neq (A_i, r_i, s_i)$ for $i \in \{1, \ldots, q\}$.
– **Type-2 Forger:** an adversary that outputs a forgery where $(A, r, s) = (A_i, r_i, s_i)$ for some $i \in \{1, \ldots, q\}$ and $(m'_1, \ldots, m'_n) \neq (m_1^i, \ldots, m_n^i)$.

We show that any forger can be used to forge $\mathsf{MAC_{BB}}$ tags. The reduction works differently for each forger type. Therefore, $\mathcal{B}$ initially chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will emulate.

• If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its $\mathsf{MAC_{BB}}$ challenger, denoted by $\mathcal{C}$, the public parameters (g_0, g_1, g, h) as well as the public key $Y = g_0^y$. Then, $\mathcal{B}$ constructs the public parameters for $\mathcal{A}$ as follows: for $i \in \{2, \ldots, n\}$, $\mathcal{B}$ chooses $\alpha_i \in_R \mathbb{Z}_p^*$ and computes $g_i = g_1^{\alpha_i}$. The parameters g_0, g_1, g, h and Y are the same as those sent by $\mathcal{C}$. $\mathcal{B}$ can answer $\mathcal{A}$'s requests as follows:

– $\mathcal{O}$`Verify` requests: when $\mathcal{A}$ sends a verify request to $\mathcal{B}$ on (A, r, s) and a block of messages $(m_1, \ldots, m_n)$, $\mathcal{B}$ computes $M = m_1 + \alpha_2 m_2 + \ldots + \alpha_n m_n$. Then, it queries its $\mathsf{MAC_{BB}}$ `Verify` oracle on (A, r, s, M) and outputs the oracle's answer to $\mathcal{A}$.
– $\mathcal{O}$`MAC` requests: when $\mathcal{A}$ sends a tag request to $\mathcal{B}$ on the block of messages $(m_1, \ldots, m_n)$, $\mathcal{B}$ asks the $\mathsf{MAC_{BB}}$ oracle on $M = m_1 + \alpha_2 m_2 + \ldots + \alpha_n m_n$. Thus, $\mathcal{B}$ obtains the tag (A_i, r_i, s_i). It sends back (A_i, r_i, s_i) to $\mathcal{A}$ which is a valid $\mathsf{MAC_{BB}^n}$ tag on $(m_1, \ldots, m_n)$.

Eventually, $\mathcal{A}$ outputs his forgery (A, r, s) on the block of messages $(m_1, \ldots, m_n)$. Using these values, $\mathcal{B}$ directly outputs its $\mathsf{MAC_{BB}}$ forgery (A, r, s) on $M' = m_1 + \alpha_2 m_2 + \ldots + \alpha_n m_n$. Therefore, $\mathcal{B}$ breaks the unforgeability of $\mathsf{MAC_{BB}}$ with the same advantage as $\mathcal{A}$.

• If $c_{mode} = 2$: In this case, $\mathcal{A}$ is rather used as a subroutine to construct a reduction $\mathcal{B}$ against the DL problem. $\mathcal{B}$ receives as input from its DL challenger, denoted by $\mathcal{C}$, the challenge $(g, H = g^v)$. The goal of $\mathcal{B}$ consists in finding v. For that purpose, it first randomly chooses $(y, g_0, h) \in_R \mathbb{Z}_p \times \mathbb{G}^2$ and computes $Y = g_0^y$. Then, it chooses $I \in \{1, \ldots, n\}$ and $(n-1)$ random values $\alpha_i \in \mathbb{Z}_p^*$. It computes, for $i \neq I, g_i = g^{\alpha_i}$ and defines $g_I = H$. $\mathcal{B}$ can answer $\mathcal{A}$'s requests as follows:

– $\mathcal{O}$`Verify` requests: when $\mathcal{A}$ sends a verify request to $\mathcal{B}$ on (A, r, s) and a block of messages $\vec{m} = (m_1, \ldots, m_n)$, $\mathcal{B}$ computes $\tilde{A} = (g_1^{m_1} \ldots g_n^{m_n} g^s \cdot h)^{\frac{1}{y+r}}$. It can thus check the validity of the quadruple $(A, r, s, \vec{m})$ by verifying whether $\tilde{A} \stackrel{?}{=} A$;
– $\mathcal{O}$`MAC` requests: as it holds y, $\mathcal{B}$ can generate a valid MAC (A, r, s) on any queried block of messages $(m_1, \ldots, m_n)$. Indeed, it chooses $r, s \in_R \mathbb{Z}_p^*$ and computes $A = (g_1^{m_1} \ldots g_n^{m_n} g^s \cdot h)^{\frac{1}{y+r}}$.

Eventually, $\mathcal{A}$ outputs his forgery (A, r, s) on a block of messages $\overrightarrow{m} = (m_1, \ldots, m_n)$. By assumption, (A, r, s) is equal to one of $\mathcal{A}$'s requests, let say (A_i, r_i, s_i), but it is a forgery on a new block of messages. Therefore, $(m_1^i, \ldots, m_n^i) \neq (m_1, \ldots, m_n)$ (one can easily show that there is at least one difference in the two blocks of messages). So with probability $\frac{1}{n}, m_I^i \neq m_I$. Thus, since $(A, r, s) = (A_i, r_i, s_i)$, we have $g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} = g_1^{m_1^i} g_2^{m_2^i} \ldots g_n^{m_n^i}$. So, the discrete logarithm v of $H = g_I$ in the base g is equal to: $v = \sum_{j \neq I}^{n} \alpha_j \frac{(m_j - m_j^i)}{m_I^i - m_I}$. Therefore, $\mathcal{B}$ can find v with probability $\frac{\varepsilon}{n}$ where ε is the probability that $\mathcal{A}$ breaks the unforgeability of $\mathsf{MAC}_{\mathsf{BB}}^n$. If $\mathcal{B}$ can break the DL problem then, it can break the $\mathsf{MAC}_{\mathsf{BB}}$ scheme (by finding the discrete logarithm of Y in the base g_0).

We can guess which of the two forgers a particular adversary $\mathcal{A}$ is with probability 1/2. So, assuming the most pessimistic scenario (case 2), $\mathcal{B}$ can break the unforgeability of $\mathsf{MAC}_{\mathsf{BB}}$ with probability $\varepsilon/2n$.

B Security Proofs of Theorem 3

Relying on the KVAC security model provided in [15], we focus in this appendix on the security proofs of our KVAC system. Owing to the lack of space, we only detail the proofs of unforgeability and anonymity.

Unforgeability. Here, we prove unforgeability when $\mathcal{A}$ is given credentials generated by the `BlindIssue` protocol. We have shown (see Theorem 2) that $\mathsf{MAC}_{\mathsf{BB}}^n$ is unforgeable under the gap $q - \mathsf{SDH}$ assumption.

Suppose there exists an adversary $\mathcal{A}$ who can break the unforgeability property of our anonymous credentials system. We will show that $\mathcal{A}$ can be used to construct an algorithm $\mathcal{B}$ that breaks the unforgeability of $\mathsf{MAC}_{\mathsf{BB}}^n$. $\mathcal{B}$ receives $pp = (\mathbb{G}, p, g_1, \ldots, g_n, g, h, g_0)$ from its $\mathsf{MAC}_{\mathsf{BB}}^n$ challenger along with Y, the issuer's public key. It sends pp and Y to $\mathcal{A}$ and answers his requests as follows:

- When $\mathcal{A}$ queries the $\mathcal{O}$`BlindIssue` oracle: $\mathcal{A}$ sends C_m and gives a proof π_1. If π_1 is invalid, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ runs the proof of knowledge extractor to extract $\{m_i\}_{i=1}^n$ and s. $\mathcal{B}$ then queries its $\mathsf{MAC}_{\mathsf{BB}}^n$ oracle on $\{m_i\}_{i=1}^n$ which returns a tag (A, r, s_u) to $\mathcal{B}$. Finally, $\mathcal{B}$ simulates the corresponding proof[6] π_2 and forwards the tag $(A, r, s_u - s)$ along with π_2 to $\mathcal{A}$.
- When $\mathcal{A}$ queries the $\mathcal{O}$`ShowVerify` oracle: $\mathcal{A}$ sends B_0, C, E along with a proof π_3. If the proof π_3 is invalid, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ runs the proof of knowledge extractor to extract $\alpha, \beta, \lambda, \delta_1, \delta_2, \ldots, \delta_{n+1}, \gamma$ and θ. If $\alpha = 0$, $\mathcal{B}$ returns 0 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ computes $A = B_0^{\alpha}$, $r = -\frac{\lambda}{\alpha}$ and $s = \delta_{n+1}$. Finally, it queries its `Verify` oracle with $((\delta_1, \delta_2, \ldots, \delta_n), (A, r, s))$ as input and returns the result to $\mathcal{A}$.

In the final `Show` protocol, $\mathcal{B}$ again extracts $\alpha, \beta, \lambda, \delta_1, \delta_2, \ldots, \delta_{n+1}, \gamma, \theta$ and outputs $((\delta_1, \delta_2, \ldots, \delta_n), (B_0^{\alpha}, -\frac{\lambda}{\alpha}, \delta_{n+1}))$ as its forgery.

[6] Such a proof can be easily simulated in the ROM, using standard techniques.

First, note that $\mathcal{B}$'s response to $\mathcal{O}$`BlindIssue` queries are identical to the ones of the honest $\mathcal{O}$`BlindIssue` algorithm. Then, we argue that its response to `ShowVerify` queries are also, with overwhelming probability, identical to the output of a real `ShowVerify` algorithm. To see this, note that the proof of knowledge property guarantees that the extractor succeeds in producing a valid witness with all but negligible probability. Furthermore, if the extractor gives valid $(\alpha, \beta, \lambda, \delta_1, \delta_2, \dots, \delta_{n+1})$, we have from $E = C^\alpha f^\beta = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \cdot B_0^\lambda \cdot f^\beta$ that

$$C^\alpha = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \cdot B_0^\lambda \Longrightarrow C^\alpha B_0^{-\lambda} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h$$

If the $\mathsf{MAC}_{\mathsf{BB}}^n$ `Verify` oracle outputs 1 on input $((\delta_1, \delta_2, \dots, \delta_n), (B_0^\alpha, -\frac{\lambda}{\alpha}, \delta_{n+1}))$, this implies that

$$\begin{aligned} & (B_0^\alpha)^{y-\frac{\lambda}{\alpha}} &&= g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \\ \Leftrightarrow\ & (B_0^\alpha)^y \cdot B_0^{-\lambda} &&= g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} \cdot h \\ \Leftrightarrow\ & (B_0^\alpha)^y \cdot B_0^{-\lambda} &&= C^\alpha B_0^{-\lambda} \\ \Leftrightarrow\ & (B_0^\alpha)^y &&= C^\alpha \\ \Leftrightarrow\ & B_0^y &&= C \end{aligned}$$

Note that α is necessarily different from 0, otherwise $B_0^\alpha = 1$ and would have been rejected by the $\mathsf{MAC}_{\mathsf{BB}}^n$ `Verify` oracle.

Thus, the honest verifier algorithm accepts, if and only if, $(B_0^\alpha, -\frac{\lambda}{\alpha}, \delta_{n+1})$ would be accepted by the $\mathsf{MAC}_{\mathsf{BB}}^n$ `Verify` algorithm for message $(\delta_1, \dots, \delta_n)$. Similarly, we can argue that $\mathcal{B}$ can extract a valid MAC from the final `Show` protocol whenever $\alpha \neq 0$ and `ShowVerify` would have output 1. Thus, if $\mathcal{A}$ can cause `ShowVerify` to accept for some statement ϕ that is not satisfied by any of the attributes sets queried to $\mathcal{O}$`BlindIssue`, then $\mathcal{B}$ can extract a new message $(\delta_1, \dots, \delta_n)$ and a valid tag for that message.

Anonymity. Suppose the user is trying to prove that he has a credential for attributes satisfying some statement ϕ. We want to show that there exists an algorithm `SimShow` that, for the adversary $\mathcal{A}$, is indistinguishable from `Show` but that only takes as input the statement ϕ and the secret key sk.

Let $\phi \in \Phi$ and $(m_1, \dots, m_n) \in \mathcal{U}$ be such that $\phi(m_1, \dots, m_n) = 1$. Let pp be the system public parameters, Y the issuer's public key and σ be such that `CredVerify`$(sk, \sigma, (m_1, \dots, m_n)) = 1$. So, σ consists of a quadruple $(A, r, s_u, \tilde{C}_m) \in \mathbb{G} \times \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{G}$ satisfying $A^{y+r} = g_1^{m_1} \dots g_n^{m_n} g^{s_u} h$.

`SimShow`(sk, ϕ) behaves as follows: it chooses a random value $l' \in_R \mathbb{Z}_p^*$ as well as a random generator $E \in_R \mathbb{G}$. It then computes $B_0 = g_0^{l'}$ and $C = Y^{l'}$. It runs $\mathcal{A}$ with the values (B_0, C, E) as the first message, simulates the proof of knowledge π_3 and outputs whatever $\mathcal{A}$ outputs at the end of the proof.

Let us first show that the values B_0, C and E are distributed identically to those produced by `Show`. Note that since $A \neq 1$, there exists $x \in \mathbb{Z}_p$ such that $A = g_0^x$. For a random value $l \in_R \mathbb{Z}_p$, $B_0 = A^l = g_0^{lx} = g_0^{l'}$ for $l' = lx$. Therefore, we also have $C = A^{ly} = Y^{l'}$. Moreover, there exists t such that $E = C^{1/l} f^t$. Then, the values computed by `SimShow` are identical to those that the normal `Show` protocol would have produced. Owing to the zero-knowledge property of the proof of knowledge, we conclude that the resulting view is indistinguishable from that produced by the adversary interacting with `Show`.

C ZKPK π_3 - Proof of Possession of a Credential

We describe an instantiation of our presentation protocol using non-interactive Schnorr-like proofs. As in [15], our protocol does not include proofs of any additional predicates ϕ, but outputs a commitment H on the attributes which may be used as input to further proof protocols.

Hereinafter, we detail the ZKPK $\pi_3 = \text{PoK}\{\alpha, \beta, \lambda, \delta_1, \dots, \delta_{n+1}, \gamma, \theta : E = C^\alpha f^\beta \wedge H = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} B_0^\lambda f^\beta \wedge C = E^\theta f^\gamma\}$ where $E = C^{1/l} f^t$, $H = E \cdot h^{-1} = g_1^{m_1} g_2^{m_2} \dots g_n^{m_n} g^{s_u} B_0^{-r/l} f^t$ and $C = E^l f^{-tl}$.

Prover		**Verifier**
Private Input: $\overrightarrow{m} = (m_1, \dots, m_n)$, l, t s_u and r		
Choose $a_1, a_2, \dots, a_{n+6} \overset{R}{\leftarrow} \mathbb{Z}_q^*$		
Compute $t_1 \leftarrow C^{a_1} f^{a_2}$		
$t_2 \leftarrow g_1^{a_3} g_2^{a_4} \dots g_n^{a_{n+2}} g^{a_{n+3}} B_0^{a_{n+4}} f^{a_2}$		
$t_3 \leftarrow E^{a_{n+5}} f^{a_{n+6}}$		
Compute $c = \mathcal{H}(Ch, t_1, t_2, t_3)$	$\xleftarrow{Ch}$	**Choose** $Ch \in_R \mathbb{Z}_p^*$
Compute $R_1 \leftarrow a_1 + c/l$, $R_2 \leftarrow a_2 + ct$	$\xrightarrow{c, R_1, \dots, R_{n+6}}$	**Compute** $t'_1 = C^{R_1} f^{R_2} E^{-c}$
for $i \in \{1, \dots, n\}$, $R_{i+2} \leftarrow a_{i+2} + cm_i$		$t'_2 = g_1^{R_3} \dots g_n^{R_{n+2}} g^{R_{n+3}} B_0^{R_{n+4}} f^{R_2} H^{-c}$
$R_{n+3} \leftarrow a_{n+3} + cs_u$, $R_{n+4} \leftarrow a_{n+4} - \frac{cr}{l}$		$t'_3 = E^{R_{n+5}} f^{R_{n+6}} C^{-c}$
$R_{n+5} \leftarrow a_{n+5} + cl$, $R_{n+6} \leftarrow a_{n+6} - ctl$		**Check if** $c = \mathcal{H}(Ch, t'_1, t'_2, t'_3)$

Proof. Let us prove that, when $C = B_0^y$, π_3 is a ZKPK of a $\mathsf{MAC}_{\mathsf{BB}}^n$ (A, r, s_u) on a block of messages $(m_1, \dots, m_n)$. The *completeness* of the protocol follows by inspection. The *soundness* follows from the extraction property of the underlying proof of knowledge[7]. In particular, the extraction property implies that for any prover $\mathcal{P}^*$ that convinces $\mathcal{V}$ with probability ε, there exists an extractor which interacts with $\mathcal{P}^*$ and outputs $(\alpha, \beta, \lambda, \delta_1, \dots, \delta_{n+1}, \gamma, \theta)$ with probability $poly(\varepsilon)$. Moreover, if we assume that the extractor inputs consists of two transcripts *i.e.* $(\mathbb{G}, g, h, f, B_0, C, E, c, \tilde{c}, R_1, \dots, R_{n+6}, \tilde{R}_1, \tilde{R}_2, \dots, \tilde{R}_{n+6})$, the witness can be obtained by computing $\alpha = \frac{R_1 - \tilde{R}_1}{c - \tilde{c}}$; $\beta = \frac{R_2 - \tilde{R}_2}{c - \tilde{c}}$; $\delta_i = \frac{R_{i+2} - \tilde{R}_{i+2}}{c - \tilde{c}}, \forall i \in \{1, \dots, n\}$; $\delta_{n+1} = \frac{R_{n+3} - \tilde{R}_{n+3}}{c - \tilde{c}}$; $\lambda = \frac{R_{n+4} - \tilde{R}_{n+4}}{c - \tilde{c}}$; $\theta = \frac{R_{n+5} - \tilde{R}_{n+5}}{c - \tilde{c}}$, $\gamma = \frac{R_{n+6} - \tilde{R}_{n+6}}{c - \tilde{c}}$; (all the computations are done mod p). The extractor succeeds when $(c - \tilde{c})$ is invertible in $\mathbb{Z}_p$. We know that $H = E \cdot h^{-1} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} B_0^\lambda f^\beta$

[7] For concurrent security, we could use the Dåmgard protocol [18] which converts any Σ protocol into a three-round interactive ZKPK secure under concurrent composition.

so $E = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} h$. We also know that $E = C^{\alpha} f^{\beta}$ so $C^{\alpha} f^{\beta} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} f^{\beta} h$ and then

$$C^{\alpha} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} h. \tag{1}$$

Since $C = B_0^y$, we have $B_0^{\alpha y} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} B_0^{\lambda} h$ and

$$B_0^{\alpha y - \lambda} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} h. \tag{2}$$

If $\boldsymbol{\alpha \neq 0}$, (2) implies that

$$(B_0^{\alpha})^{y - \frac{\lambda}{\alpha}} = g_1^{\delta_1} \dots g_n^{\delta_n} g^{\delta_{n+1}} h. \tag{3}$$

Let $A = B_0^{\alpha}$, $r = -\frac{\lambda}{\alpha}$, $s_u = \delta_{n+1}$ and $m_i = \delta_i$ for $i \in \{1, \dots, n\}$.

If $\alpha \neq 0$, (3) implies that the prover knows a valid $\mathsf{MAC}_{\mathsf{BB}}^n$, (A, r, s_u) on a block of messages $(m_1, \dots, m_n)$. Note that $y - \frac{\lambda}{\alpha} \neq 0$, otherwise this would imply that the prover knows y which would be equal to $\frac{\lambda}{\alpha}$.

Let us now prove that $\alpha \neq 0$. We know that

$$C = E^{\theta} f^{\gamma} = (C^{\alpha} f^{\beta})^{\theta} f^{\gamma} = C^{\alpha\theta} f^{\beta\theta + \gamma} \Longrightarrow 1 = C^{\alpha\theta - 1} f^{\beta\theta + \gamma} \tag{4}$$

• If the prover does not know the discrete logarithm of C in the base f, this implies that it only knows one representation $(0, 0)$ of 1 in the base (C, f) [8]. Therefore, $\alpha\theta = 1$ which implies that $\alpha \neq 0$.

• Suppose now that the prover knows the discrete logarithm χ of C in the base f (*i.e.* $C = f^{\chi}$) and that $\alpha = 0$. Since $C = B_0^y$, we have $B_0^y = f^{\chi}$ and then $B_0 = f^{\frac{\chi}{y}}$ (since $Y = g_0^y \neq 1$, this implies that $y \neq 0 \mod p$). From (1) and since α is supposed to be equal to 0, we have that $h = g_1^{-\delta_1} g_2^{-\delta_2} \dots g_n^{-\delta_n} g^{-\delta_{n+1}} f^{-\lambda \frac{\chi}{y}}$.

So, the issuer could use the prover as a subroutine to compute a representation of h in the base $(g_1, g_2, \dots, g, f)$. As $(g_1, g_2, \dots, g, f)$ are random generators of $\mathbb{G}$, this is impossible under the DL assumption [8]. Therefore, this means that either $\mathcal{P}^*$ does not know the discrete logarithm of C in the base f or $\alpha \neq 0$. Both cases imply that $\alpha \neq 0$. We therefore conclude that $\alpha \neq 0$ and so the prover knows a valid $\mathsf{MAC}_{\mathsf{BB}}^n$ (A, r, s_u) on a block of messages $(m_1, \dots, m_n)$.

Finally, to prove (honest-verifier) *zero-knowledge*, we construct a simulator `Sim` that will simulate all interactions with any (honest verifier) $\mathcal{V}^*$.

1. `Sim` randomly chooses $l' \in_R \mathbb{Z}_p^*$ and a random generator $E \in_R \mathbb{G}$ and then computes $B_0 = g^{l'}$ and $C = Y^{l'}$.
2. `Sim` randomly chooses $c, R_1, \dots, R_{n+6} \in_R \mathbb{Z}_p^*$ and computes $t_1 = C^{R_1} f^{R_2} E^{-c}$, $t_2 = g_1^{R_3} \dots g_n^{R_{n+2}} g^{R_{n+3}} B_0^{R_{n+4}} f^{R_2} H^{-c}$ and $t_3 = E^{R_{n+5}} f^{R_{n+6}} C^{-c}$.
3. `Sim` outputs $S = \{B_0, C, E, c, R_1, R_2, \dots, R_{n+6}\}$.

Since $\mathbb{G}$ is a prime-order group, then the blinding is perfect in the first step. Indeed, there exists $x \in \mathbb{Z}_p$ such that for a valid $\mathsf{MAC}_{\mathsf{BB}}^n$ (A, r, s_u) on $(m_1, \dots, m_n)$: $A = g_0^x$.

For a random value $l \in \mathbb{Z}_p^*$, we therefore have $B_0 = A^l = g_0^{lx} = g_0^{l'}$ for $l' = lx$. This also implies that $C = A^{ly} = Y^{l'}$. Moreover, there exists t such that $E = C^{\frac{1}{l}} f^t$. Therefore S and $\mathcal{V}^*$'s view of the protocol are statistically indistinguishable.

References

1. Abe, M.: A secure three-move blind signature scheme for polynomially many signatures. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 136–151. Springer, Heidelberg (2001). doi:10.1007/3-540-44987-6_9
2. Akagi, N., Manabe, Y., Okamoto, T.: An efficient anonymous credential system. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 272–286. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85230-8_25
3. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: CCS 2013, pp. 1087–1098. ACM (2013)
4. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_33
5. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_4
6. Brands, S.: Untraceable off-line cash in wallet with observers. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 302–318. Springer, Heidelberg (1994). doi:10.1007/3-540-48329-2_26
7. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_22
8. Brands, S.A.: An efficient off-line electronic cash system based on the representation problem. Technical report CS-R9323. CWI, Amsterdam (1993)
9. Camenisch, J., Dubovitskaya, M., Haralambiev, K., Kohlweiss, M.: Composable and modular anonymous credentials: definitions and practical constructions. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 262–288. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48800-3_11
10. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001). doi:10.1007/3-540-44987-6_7
11. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003). doi:10.1007/3-540-36413-7_20
12. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_4
13. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical report (1997)
14. Canard, S., Coisel, I., Jambert, A., Traoré, J.: New results for the practical use of range proofs. In: Katsikas, S., Agudo, I. (eds.) EuroPKI 2013. LNCS, vol. 8341, pp. 47–64. Springer, Heidelberg (2014). doi:10.1007/978-3-642-53997-8_4
15. Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic MACs and keyed-verification anonymous credentials. In: CCS 2014, pp. 1205–1216. ACM (2014)
16. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (1985)
17. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993). doi:10.1007/3-540-48071-4_7

18. Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_30
19. Fuchsbauer, G., Hanser, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 233–253. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48000-7_12
20. Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Transferable constant-size fair e-cash. Cryptology ePrint Archive, Report 2009/146 (2009). http://eprint.iacr.org/
21. Hinterwälder, G., Zenger, C.T., Baldimtsi, F., Lysyanskaya, A., Paar, C., Burleson, W.P.: Efficient e-cash in practice: NFC-based payments for public transportation systems. In: Cristofaro, E., Wright, M. (eds.) PETS 2013. LNCS, vol. 7981, pp. 40–59. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39077-7_3
22. IBM: Specification of the identity mixer cryptographic library (revised version 2.3.0). IBM Research Report RZ 3730 (2010). http://domino.research.ibm.com/library/cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/eeb54ff3b91c1d648525759b004fbbb1?OpenDocument
23. Paquin, C.: U-Prove Technology Overview V1.1 (Revision 2). In: Microsoft Technical report (2013). http://research.microsoft.com/pubs/166980/U-ProveTechnologyOverviewV1.1Revision2.pdf
24. Paquin, C., Zaverucha, G.: U-Prove cryptographic specification V1.1 (Revision 3). In: Microsoft Technical report (2013). http://research.microsoft.com/pubs/166969/U-ProveCryptographicSpecificationV1.1.pdf
25. de la Piedra, A., Hoepman, J.-H., Vullers, P.: Towards a full-featured implementation of attribute based credentials on smart cards. In: Gritzalis, D., Kiayias, A., Askoxylakis, I. (eds.) CANS 2014. LNCS, vol. 8813, pp. 270–289. Springer, Cham (2014). doi:10.1007/978-3-319-12280-9_18
26. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
27. Vullers, P., Alpár, G.: Efficient selective disclosure on smart cards using idemix. In: Fischer-Hübner, S., de Leeuw, E., Mitchell, C. (eds.) IDMAN 2013. IAICT, vol. 396, pp. 53–67. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37282-7_5