Computer and Information Science; Vol. 15, No. 2; 2022
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Efficient and Traceable Anonymous Credentials on Smart Cards

Wei Wu[1]

[1] College of Cyber Security, Jinan University, Guangzhou, China

Correspondence: Wei Wu, College of Cyber Security, Jinan University, Guangzhou, China.

Received: Janaury 24, 2021 Accepted: March 10, 2022 Online Published: March 17, 2022

doi:10.5539/cis.v15n2p58 URL: https://doi.org/10.5539/cis.v15n2p58

Abstract

Anonymous credential (AC) systems allow users, obtaining a credential on a set of attributes, to anonymously prove ownership of the credential and then to selectively disclose a subset of attributes without leaking any other attributes. Recently, a new type of AC, called keyed-verification anonymous credential (KVAC), has been proposed, which indicates that the credential issuer is also the verifier. Conceptually, the KVAC system is suitable for being used as employee cards, library access cards or eIDs (electronic ID cards). However, since the limited process power of smart cards, most of the existing KVAC systems are hard to be implemented on them. In addition, none of the existing KVAC systems provide traceability to obtain the user's identity if anyone tries to misbehave with KVAC. In this paper, we present the first efficient and traceable KVAC system designated for smart cards. Our scheme provides the following security properties: unforgeability, anonymity, traceability and unlinkability. To demonstrate the efficiency and feasibility, we present an implementation of our scheme on standard Multos smart cards. The implementation results show that our scheme is efficient enough for practical use.

Keywords: privacy-preservation, anonymous credentials, traceability, smart cards

1. Introduction

Anonymous credential (AC) systems, proposed by Chaum (1985) and formalized by Camenisch and Lysyanskaya (2001), are one of privacy-preserving technologies supporting anonymous authentications for users and ensuring accountability for verifiers. In an AC system, a user, obtaining a credential on a set of attributes (e.g., name, age, nationality, or the validity of a pre-paid pass) from a credential issuer, can anonymously prove ownership of his credential and then selectively disclose a subset of attributes (e.g., only age) without leaking any other attributes. Informally, an AC is a public key signature scheme, signed by an issuer, on a set of attributes and the user's public key. Therefore, the user is the only one who can use the credential and prove a subset of attributes with a zero-knowledge proof of the corresponding secret key. Recently, Chase, Meiklejohn, & Zaverucha (2014) introduced a new type of AC, called keyed-verification anonymous credential (KVAC), which indicates that the credential issuer is also the verifier. In these settings, message authentication code (MAC) of symmetric key techniques is used instead of public key signatures as the main building blocks for KVAC systems. Conceptually, the KVAC systems are suitable for being designed as employee cards, library access cards or eIDs (electronic ID cards). For example, library access cards are issued by a library and are usually used within the library, in which both the credential issuer and the verifier are the same entity.

However, there is a lack of practical implementations of KVAC on smart cards. The two main problems are as follows. First, the limited processing power of smart cards poses a huge challenge for running complex cryptographic protocols or advanced cryptography, even for modern smart cards. Consequently, implementing KVAC on smart cards involves a trade-off between the design of cryptographic schemes and the capabilities of smart cards. Second, most available smart cards do not provide API for a bilinear map, which is a very popular operation for AC systems and KVAC systems. Moreover, simple operations over an elliptic curve, such as elliptic curve point scalar multiplication or addition, are severely restricted due to the requirement of the running time for authentications. Therefore, a KVAC system designed for smart cards needs to avoid the operation of the bilinear map and minimize the operation of elliptic curve point scalar multiplication or addition.

When KVAC is deployed on smart cards, there is another issue that we have to consider. KVAC systems provide anonymity for users to prove the ownership of credentials without leaking their identity information, which is a double-edged sword that can also be used by criminals to engage in unlawful activities. If the user misbehaves

with KVAC, it should be possible to make the user responsible for his illegal actions. Consequently, it is preferable for KVAC systems to provide traceability to de-anonymize misbehaving users without compromising the anonymity of other honest users. However, none of the existing KVAC systems provide traceability.

In this paper, we address the above challenges: First, we proposed the first efficient and traceable KVAC systems designed for smart cards. Our scheme simultaneously provides the following security properties:

- **Unforgeability**: Users cannot successfully prove ownership of credentials that do not belong to them, nor can they successfully disclose attributes that were not previously authenticated.
- **Anonymity**: The user's identity information is not revealed during a showing of his credential.
- **Traceability**: A tracing authority can de-anonymize misbehaving users' identity information.
- **Unlinkability**: No verifiers can distinguish whether the showing credential belongs to the same user, even if the same attributes are disclosed multiple times.

Second, we develop an implementation of our KVAC on a standard Multos smart card and measure the total time cost for the computation and communication on the smart card. Then, the implementation results show that our scheme is efficient enough for practical use.

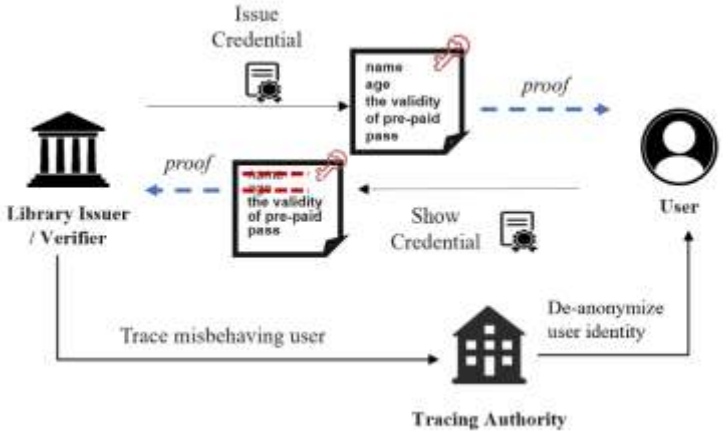

Figure 1. Example Scenario of Our KVAC System

Example Scenario. As shown in Figure 1, we chose a library scenario to illustrate the practicality of our KVAC on smart cards. In a municipal library, users can borrow books as long as they pay a fee every quarter. A user can be issued a library access card as a credential, which certified the validity of a pre-paid pass and other identity attributes. Then, the user is able to prove ownership of the card and only selectively disclose his validity of the pre-paid pass to the library manager. By using the library access card, a user can rent books without leaking other irrelevant attributes such as age and nationality. Then, the user just simply waves his contactless library access card when he leaves the municipal library with books. The first security property, *unforgeability*, assures that nobody can prove ownership of a library access card that does not belong to him or try to rent books without paying the pre-paid fee. The *anonymity* property protects users' privacy, such as his reading habits due to nobody can link the identity information from the books he reads. The *traceability* property allows the library to trace and identify readers who violate the rules of the library, such as stealing or polluting books. Finally, the *unlinkability* property prevents the profiling of users. All reading records of a user are unlinkable, and the library cannot check which books users read subsequently.

Organization. The remainder of this paper is organized as follows. In Section 2, we discuss existing related work. Section 3 presents the preliminaries of our KVAC system. We detailed the specific construction in Section 4. In Section 5, we describe our implementation on a smart card and discuss the experiment results.We conclude this paper in Section 6.

2. Related Work

Cryptographic anonymous credential (AC) systems were proposed by Chaum (1985) and formalized by Camenisch and Lysyanskaya (2001). After years of research, AC systems were gradually improved by using

more efficient cryptographic primitives (Fuchsbauer, Hanser, & Slamanig, 2019; Sanders, 2020) and developing formal security models and security analysis (Camenisch et al., 2014; Tan & Groß, 2020). Up to date, Idemix (Bichsel et al, 2009) and U-Prove (Paquin & Zaverucha, 2011) are the well-known applications of AC systems, which both aim for practical use.

Furthermore, a new approach to obtain more efficient AC systems was proposed. Chase et al., (2014) discovered that the credential issuer can be served as the verifier in many scenarios. This means that the verifier also owns the issuer key pair, which can be used to construct more efficient AC systems. They then formally defined these as keyed-verification anonymous credential (KVAC) systems and provided concrete instantiation.

Afterward, many new constructions of KVAC had been proposed. Assume a user possesses a credential on n attributes and wants to selectively only k attributes for a showing. The KVAC system proposed by Barki, Brunet, Desmoulins, & Traoré (2016) needs to cost $n\text{-}k+12$ exponentiations, while the KVAC system constructed by Couteau and Reichle (2019) needs to cost $2n\text{-}2k+3$ exponentiations. Although their KVAC systems are efficient enough on the PC platform, they cannot be directly adapted to the smart cards platform.

The first KVAC system deployed on the smart cards was proposed by Camenisch, Drijvers, Dzurenda, & Hajny (2019), which achieves the most efficient proving protocol with $n\text{-}k+2$ exponentiations to prove ownership of a credential. Moreover, all the operations of their KVAC can be achieved by basic arithmetic operations which are already provided by modern smart cards' API. Their KVAC system is suitable for being implemented on smart cards. However, traceability is not within their design goals, making it very easy for any user misbehaves with their KVAC systems, especially on smart cards.

Recently, Chase, Perrin, & Zaverucha (2020) construct a new KVAC based on an algebraic MAC instantiated in a cyclic group G. The advantage is that attributes can be elements in G, whereas the previous KVAC only supports attributes that were integers of a multiplicative group Z_p. Therefore, we can encrypt group elements using Elgamal encryption, and prove that the encrypted elements are certified by a credential in zero-knowledge. Because the algebraic MAC, encryption and the zero-knowledge proofs are all instantiated in G, it only needs basic arithmetic operations that are already supported by modern smart cards, which means that their scheme is suitable deployed on smart cards. However, their KVAC system still does not consider the security of traceability. Obviously, it is still a problem to be solved that how to achieve an efficient and traceable KVAC on smart cards.

3. Preliminaries

In this section, we will first introduce the basic notations of our KVAC system. Then, we give a brief overview of some cryptographic building blocks in terms of algebraic MAC, zero-knowledge proofs and Elgamal Encryption. At last, we briefly introduce the MULTOS smart cards.

3.1 Algebraic MAC

Traditional MACs are constructed using symmetric key primitives like HMAC (Krawczyk, Bellare, & Canetti, 1997) based on hash functions and Poly1305-AES (Bernstein, 2005) based on block ciphers. However, these traditional MACs cannot combine efficient zero-knowledge proofs. To address this problem, Dodis, Kiltz, Pietrzak, & Wichs (2012) proposed algebraic MACs, which are constructed using group operations and support efficient zero-knowledge proofs. Then, Chase et al., (2014) showed that certain algebraic MACs can be used as building blocks for constructing KVAC systems, and they instantiated the first KVAC system.

In this paper, we use the algebraic MAC proposed by Camenisch et al., (2019) as the underlying building block to construct our efficient and traceable KVAC systems, which is primarily designed for smart cards.

We denote this algebraic MAC scheme with following algorithms {*Setup, MACKeyGen, MAC, Verify*}:

$$Setup\ (\lambda) \rightarrow\ par$$

Input a security parameter λ. Choose a random generator g from cyclic group G of prime order p, i.e., $g \in G$. Output the public parameters $par = (G, g, p)$.

$$MACKeyGen(par, n) \rightarrow\ (key, ipar)$$

Input the public parameters $par = (G, g, p)$ and an integer n defining an upper bound on the number of messages in a MAC. Choose $n+1$ random integer $(x, x_1, \ldots, x_n)$ from multiplicative group Z_p. Output the MAC secret key $key = (x, x_1, \ldots, x_n)$ and the issuer parameters $(X, X_1, \ldots, X_n) = (g^{x_0}, g^{x_1}, \ldots, g^{x_n})$.

$$MAC(par, key, \boldsymbol{M}) \rightarrow \sum = (\sigma, \sigma_{x_1}, \ldots, \sigma_{x_n})$$

Input the public parameters $par = (G, g, p)$, the MAC secret key key and a set of messages $\boldsymbol{M} = (M_1, \ldots, M_n) \in Z_p$. Compute $\sigma = g^{1/(x+M_1 x_1+\ldots+M_n x_n)}$ and $(\sigma_{x_1}, \ldots, \sigma_{x_n}) = (\sigma^{x_1}, \ldots, \sigma^{x_n})$. Output the message authentication code $\sum = (\sigma, \sigma_{x_1}, \ldots, \sigma_{x_n})$.

$$Verify(par, key, \sum, \boldsymbol{M}) \rightarrow valid / invalid$$

Input the public parameters $par = (G, g, p)$, the MAC secret key key, the message authentication code $\sum$ and messages $\boldsymbol{M} = (M_1, \ldots, M_n)$. Output *valid* if $g = \sigma^{(x_0+M_1 x_1+\ldots+M_n x_n)}$; otherwise, output *invalid*.

3.2 Zero-knowledge Proofs

The concept of zero-knowledge proof was first introduced by Goldwasser, Micali, & Rackoff (1985). Then, multiple constructions of zero-knowledge proofs are used to prove knowledge of discrete logarithms and representations of elements in G. A zero-knowledge proofs system satisfies the following properties:

- Completeness: if the statement is true and the honest prover indeed knows the secret which satisfies the statement, then the prover can generate a proof to convince the honest verifier;
- Soundness: if the statement is false, then no cheating prover can generate a proof to convince the verifier, except with some small probability;
- Zero-knowledge: if the statement is true, no verifier learns anything other than the fact that the statement is true (the secret is not leaking);
- Proof of knowledge: there exists a probabilistic, polynomial-time (PPT) knowledge extractor that, given black-box access to a successful prover, can always compute the secret from the correct proof.

In our KVAC system, we make use of the notation introduced by Camenisch and Stadler (1997) to prove knowledge of discrete logarithms and representations of elements in G. A non-interactive zero-knowledge proof of knowledge $\boldsymbol{\pi}$ is described by:

$$\boldsymbol{\pi} = ZKPoK\ \{(x, y\ldots): statements\ about\ x, y\ldots\},$$

where (x, y…) are secrets (discrete logarithms or representations of elements in G) that satisfy the statements. In this paper, the best instantiation of zero-knowledge proof is the Schnorr-type proof (Schnorr, 1991).

3.3 Elgamal Encryption

The Elgamal encryption scheme (ElGamal, 1985) is a well-known and practical public key encryption scheme that is provably secure against adaptive chosen plaintext attacks without random oracles.

We denote the Elgamal encryption scheme with the following algorithms {*Setup*, *EncKeyGen*, *Enc*, *Dec*}:

$$Setup\ (\lambda) \rightarrow par$$

Input a security parameter λ. Choose a random generator g from cyclic group G of prime order p, i.e., $g \in G$. Output the public parameters $par = (G, g, p)$.

$$EncKeyGen(par) \rightarrow (pk, sk)$$

Input the public parameters $par = (G, g, p)$. Choose a random integer sk from a multiplicative group Z_p, i.e., $sk \in Z_p$. Output the public key $pk = g^{sk}$ and the secret key sk.

$$Enc(par, pk, m) \rightarrow CT = (ct_1, ct_2)$$

Input the public parameters $par = (G, g, p)$, the public key pk and a message m. Choose a random integer r from multiplicative group Z_p, i.e., $r \in Z_p$. Output the ciphertext $CT = (ct_1, ct_2) = (g^r, pk^r\, m)$.

$$Dec(par, sk, CT) \rightarrow m$$

Input the public parameters $par = (G, g, p)$, the secret key sk and a ciphertext c. Output the decrypted message $m = ct_2 / (ct_1)^{sk} = (pk^r\, m) / (g^{r\, sk}) = (pk^r\, m) / pk^r$.

3.4 MULTOS Smart Cards

One of the goals of our research work is to assess the efficiency of our KVAC system when it runs on a smart card. Hence, implementing a KVAC system requires an open smart card platform that provides the necessary cryptographic operations support. In practice, we choose the MULTOS smart card platforms (Torr & Mayes, 2017) for implementing our KVAC system.

The MULTOS is a multi-application operating system designed for smart card products. A highly secure and independent execution platform is run on the operating system of the MULTOS. A MULTOS smart card executes specific assembler code called MULTOS Execution Language (MEL) and provides the developers with specific interfaces for the smart card. In addition, MULTOS has a rich function library available supporting the common operations required by developers, such as PIN handling, memory management and cryptographic operations. Most of the operations that the hardware can support are possibly provided in the MEL. Due to MEL providing the full base for programming MULTOS cards, we can easily write programs and implement our KVAC system on the MULTOS smart cards.

4. Our Keyed-Verification Anoynomous Credential System

Unlike traditional anonymous credential (AC) systems, the KVAC systems are particularly suitable for scenarios where a credential issuer is the same entity as a verifier, which means that the verifier also knows the secret keys of the credential issuer. Our keyed-verification anonymous credential (KVAC) system is based on the algebraic MAC scheme described in Section 3 above. Moreover, our KVAC system supports all the standard security properties of AC systems, such as unforgeability, anonymity and unlinkability. In addition, we provide traceability to support de-anonymizing misbehaving users without compromising the anonymity of other honest users.

In this section, we present our KVAC system using the algebraic MAC as the main building block. All attributes are elements in multiplicative group Z_p and are parametrized by n denoting the maximum number of attributes in a credential. The basic idea of our KVAC is that the credential issuer assigns a unique user identifier *uid* to the user and signs *uid* and attributes $M = (m_1, \ldots, m_n)$ using the algebraic MAC. Then, the user obtains the message authentication code as his credential $\sum$. When showing a credential with disclosed attributes $\boldsymbol{M}_D$ (the disclosed attributes M_D are a subset attributes of M), the user firstly computes g^{uid} and encrypts g^{uid} under the tracing authority public key *tpk* using Elgamal encryption to generate pseudonym *nym*. He then randomizes his credential (denote it $\sum'$) and sends ($\sum'$, *nym*) with a zero-knowledge proof $\boldsymbol{\pi}$ proving that the disclosed attributes $\boldsymbol{M}_D$ are certified by $\sum'$ and the pseudonym *nym* is correctly encrypted the user identifier *uid* under the tracing authority public key by *nym*. Finally, if the verifier discovers that the user possessing *nym* misbehaves with $\sum'$, the verifier can notify the tracing authority to decrypt the *nym* using tracing authority secret key *tsk* to get the g^{uid}, which reveals the identity information of the misbehaving user.

A KVAC scheme consists of four algorithms (*Setup*, *TAKeyGen*, *CredKeyGen*, *Trace*) and two protocols ((*CredObtain*, *CredIssue*), (*CredProve*, *CredVerify*)) that are run by a user and a credential issuer who also serves as a verifier. The details of the construction are as follows:

$$Setup(\lambda) \rightarrow par \tag{1}$$

Input a security parameter λ and an integer $n \in Z_p$ defining an upper bound on the number of certified attributes in a credential. Choose a random generator g from cyclic group G of prime order p, i.e., $g \in G$ and a secure hash function $Hash$: $\{0,1\}^* \rightarrow Z_p$, which is also a random oracle in the security analysis. Output the public parameters $par = (G, g, p, Hash)$.

$$TAKeyGen(par) \rightarrow (tpk, tsk) \tag{2}$$

This algorithm *TAKeyGen* is run by a tracing authority. Input the public parameters $par = (G, g, p, Hash)$, and choose a random integer *tsk* from a multiplicative group Z_p, i.e., $tsk \in Z_p$. Output the public key $tpk = g^{tsk}$ and the secret key *tsk* for the tracing authority.

$$CredKeyGen(\lambda, n) \rightarrow (ipar, isk) \tag{3}$$

This algorithm *CredKeyGen* is run by a credential issuer/verifier. Input the public parameters $par = (G, g, p, Hash)$ and an integer n defining an upper bound on the number of certified attributes in a credential, and choose $n+2$ random integer $(x, x_0, x_1, \ldots, x_n)$ from multiplicative group Z_p. Output the issuer secret key $isk = (x, x_0, x_1, \ldots, x_n)$ and the issuer parameters $ipar = (X, X_0, X_1, \ldots, X_n) = (g^x, g^{x_0}, g^{x_1}, \ldots, g^{x_n})$.

$$(CredObtain(par, ipar, \boldsymbol{M}), CredIssue(par, isk, \boldsymbol{M})) \rightarrow (\sum, uid, nym) \tag{4}$$

The part *CredObtain* is run by a user and takes as input the public parameters $par = (G, g, p, Hash)$, the issuer parameter $ipar = (X, X_0, X_1, \ldots, X_n)$ and a set of attributes $\boldsymbol{M} = (m_1, \ldots, m_n) \in Z_p$.

The part *CredIssue* is run by a credential issuer and takes as input the public parameters *par*, the issuer secret key $isk = (x, x_0, x_1, \ldots, x_n)$ and the attributes $\boldsymbol{M}$.

At the end of the protocol, the credential issuer outputs the credential $\sum$, a unique user identifier *uid* and a pseudonym *nym* to the user. We have to mention that attributes $\boldsymbol{M}$ must be transmitted over a secure channel and

the resulting credential $\sum$, user identifier *uid* and g^{uid} should be stored by the credential issuer. We depict the whole protocol (*CredObtain*, *CredIssue*) in Table 1.

Table 1. The Protocol (*CredObtain*, *CredIssue*)

User	***Credential Issuer***
$par, ipar = (X, X_0, X_1, \ldots, X_n), \boldsymbol{M} = (m_1, \ldots, m_n)$	$par, isk = (x, x_0, x_1, \ldots, x_n), \boldsymbol{M} = (m_1, \ldots, m_n)$
	Select two random integer $uid, r \in Z_p$;
	Compute $\sum = (\sigma, \sigma_{x_1}, \ldots, \sigma_{x_n})$, where $\sigma = g^{1/(x+uid\ x_0+m_1\ x_1+\ldots+m_n\ x_n)}$
	and $(\sigma_x, \sigma_{x_0}, \ldots, \sigma_{x_n}) = (\sigma^x, \sigma^{x_0}, \ldots, \sigma^{x_n})$;
	Compute $\pi_{MAC} = \{(x, x_0, x_1, \ldots, x_n): (\sigma_x, \sigma_{x_0}, \ldots, \sigma_{x_n}) = (\sigma^x, \sigma^{x_0}, \ldots, \sigma^{x_n}) \wedge (X, X_0, X_1, \ldots, X_n) = (g^x, g^{x_0}, g^{x_1}, \ldots, g^{x_n})\}$;
	Store $\sum, \boldsymbol{M}, uid, g^{uid}$ to the issurance records;
←	Send $(\sum, uid, \pi_{MAC})$ to the user.
Check whether: $g = \sigma_x \sigma_{x_0}^{uid} \sigma_{x_1}^{m_1} \ldots \sigma_{x_n}^{m_n}$ and verify $\boldsymbol{\pi}_{MAC}$	

Note. $\boldsymbol{\pi}_{MAC}$ is used to prove that the credential issuer correctly signs the user identifier *uid* and the attributes $\boldsymbol{M} = (m_1, \ldots, m_n)$ with his secret key $isk = (x, x_0, x_1, \ldots, x_n)$ in zero-knowledge. $\boldsymbol{\pi}_{MAC}$ can be instantiated using Schnorr-type proof (Schnorr, 1991).

$$(CredProve(par, \sum, \boldsymbol{M}, D, uid, tpk), CredVerify(par, isk, D, \boldsymbol{M_D}, nym)) \rightarrow valid / invalid \quad (5)$$

The part *CredProve* is run by a user and takes as input the public parameters $par = (G, g, p)$, a credential $\sum = (\sigma, \sigma_{x_1}, \ldots, \sigma_{x_n})$, a set of attributes $\boldsymbol{M} = (m_1, \ldots, m_n) \in Z_p$, an index set D denoting which indexes in $\boldsymbol{M}$ are disclosed, the user identifier *uid* and the tracing authority public key $tpk = g^{tsk}$.

The part *CredVerify* is run by a verifier who is also the credential issuer, and takes as input the public parameters $par = (G, g, p)$, the issuer secret key *isk*, the index set D, the disclosed attributes $\boldsymbol{M}_D$ (a subset of $\boldsymbol{M}$ on index set D) and the user's pseudonym *nym*.

At the end of the protocol, the verifier outputs are *valid* for acceptance or *invalid* for rejection. We depict the whole protocol (*CredObtain*, *CredIssue*) in Table 2.

Table 2. The Protocol (*CredProve*, *CredVerify*)

User	***Verifier***
$par, \sum = (\sigma, \sigma_{x_1}, \ldots, \sigma_{x_n}), \boldsymbol{M} = (m_1, \ldots, m_n), D, uid$	$par, isk = (x, x_0, x_1, \ldots, x_n), D, \boldsymbol{M_D}, nym$
←	Select a random $nonce \in Z_p$ and send it to user
Select random $r, k, \rho_r, \rho_{uid}, \rho_k$ and ρ_{m_i} for all $i \notin D$	
Randomize credential by computing $\sum' = \sigma^r$	
Encrypt g^{uid} to get $nym = (nym_1, nym_2) = (g^k, tpk^k g^{uid})$;	
Compute $t_1 = \sigma_{x_0}^{\rho_{uid} \cdot r} \cdot \prod_{i \notin D} \sigma_{x_i}^{\rho_{m_i} \cdot r} \cdot g^r$,	
$t_2 = g^{\rho_k}$,	
and $t_3 = tpk^{\rho_k} \cdot g^{\rho_{uid}}$	
Compute $c = Hash(D, \boldsymbol{M}_D, t_1, t_2, t_3, \sigma', par, ipar, nym, nonce)$	
Compute $s_r = \rho_r + c\ r$,	
$s_k = \rho_k + c\ k$,	
$s_{uid} = \rho_{uid} + c\ uid$,	

and $s_{m_i} = \rho_{m_i} - c\, m_i$ for all $i \notin D$

Send $(\sum', t_1, t_2, t_3, s_r, s_{uid}, s_k, s_{m_i}, nym)$ ⟶

Compute $c = Hash(D, \boldsymbol{M}_D, t_1, t_2, t_3, \sigma', par, ipar, nym, nonce)$

Check whether:

$t_1 = g^{s_r} \cdot (\sum')^{-c \cdot x + x_0 \cdot s_{uid} + \sum_{i \notin D}(x_i \cdot s_{m_i}) - \sum_{i \in D}(x_i \cdot m_i \cdot c)}$,

$g^{s_k} = (nym_1)^c \cdot t_2$

and $g^{s_{uid}} \cdot (tpk)^{s_k} = (nym_2)^c \cdot t_3$

Note. This protocol is an instantiation of zero-knowledge proof $\boldsymbol{\pi} = \{(r, k, \rho_r, \rho_{uid}, \rho_k, \{\rho_{m_i}\}_{i \notin D}, \{m_i\}_{i \notin D}, uid)$: $Verify(par, isk, \sum', \boldsymbol{M}_D) = valid \wedge Enc(par, tpk, uid) = nym\}$, which prove that the disclosed attributes $\boldsymbol{M}_D$ are certified by credential $\sum'$ and the *nym* is correctly encrypted the user identifier *uid* under the tracing authority public key.

$$Trace(par, tsk, nym) \rightarrow uid \tag{6}$$

This algorithm *Trace* is run by a tracing authority. Input $par = (G, g, p, Hash)$, the tracing authority secret key *tsk* and a user's pseudonym $nym = (g^r, tpk^r\, g^{uid})$. Decrypt $g^{uid} = (tpk\, g^{uid}) / (g^{r\, sk}) = (tpk\, g^{uid}) / tpk$ and send g^{uid} to the credential issuer who then search *uid* from his inssurance records.

5. Implementation and Evaluation

In this section, we present the execution time of our KVAC system on MULTOS smart cards. For our implementation, we make use of development cards based on the SLE66 chip from Infineon. This particular MULTOS implementation supports a wide range of modulo arithmetic operations, which is sufficient to support our KVAC system on smart cards. This is the main reason why we choose MULTOS smart cards in this implementation. We mainly evaluate the execution time of the protocol (*CredProve, CredVerify*) which was implemented on a MULTOS smart card with a standard NIST P-192 curve (Kerry & Director, 2013). The total time for the whole protocol on the smart card including computation and communication cost with a terminal (standard PC, AMD Ryzen 5 3600, 6-core processor CPU, 8 GB RAM, Ubuntu 20.04) makes our implementation suitable for different application scenarios like employee cards and library access cards.

We artificially limited the number of attributes per user to a maximum of 5 which we believe is sufficient for practical use, but in fact, the available memory resources of the MULTOS smart cards will allow storing up to 50 attributes on a single card. Another limitation we implemented was caused by the limited resources of the MULTOS smart cards. We had to limit the prime number size to 1024 bits and use only the SHA-1 hash function.

As shown in Table 3, the protocol (*CredProve, CredVerify*) can almost be completed in less than 1000 milliseconds. In the worst-case scenario, there are five attributes stored on the smart card, none of which are disclosed during the protocol run, which means that all 5 attributes need to be proven in zero-knowledge. Overall, the implementation results demonstrate the efficiency and feasibility of our KVAC system on smart cards.

Table 3. Time Cost for the protocol (*CredProve, CredVerify*)

	Disclose 0 attributes	Disclose 1 attributes	Disclose 2 attributes	Disclose 3 attributes	Disclose 4 attributes	Disclose 5 attributes
Total 2 attributes in credential	820 ms	683 ms	527 ms	---	---	---
Total 3 attributes in credential	932 ms	817 ms	713 ms	602 ms	---	---
Total 4 attributes in credential	1052 ms	921 ms	812 ms	704 ms	592 ms	---
Total 5 attributes in credential	1165 ms	1026 ms	947 ms	826 ms	712 ms	600 ms

Note. Disclosing 0 attributes means a user proves ownership of his credential and proves knowledge of all attributes in zero knowledge.

6. Conclusion and Future Work

There are very few practical anonymous credential systems, and implementations on smart cards are either too slow or do not provide the traceability to de-anonymize misbehaving users. In this paper, we address this problem was twofold: 1) propose an efficient and traceable anonymous credential system on smart cards that simultaneously satisfies unforgeability, anonymity, traceability and unlinkability; 2) develop an implementation of our KVAC system on a standard MULTOS smart card, and the experimental results show the efficiency and feasibility of our scheme.

In order to meet the demands of time-critical applications, we will continue to improve the efficiency and security of our KVAC system on different devices. Furthermore, in addition to further optimization, we will combine our KVAC system with a suitable revocation solution to revoke all credentials of misbehaving users as well as lost credentials.

References

Barki, A., Brunet, S., Desmoulins, N., & Traoré, J. (2016). Improved algebraic MACs and practical keyed-verification anonymous credentials. In *International Conference on Selected Areas in Cryptography* (pp. 360-380). Springer, Cham. https://doi.org/10.1007/978-3-319-69453-5_20

Bernstein, D. J. (2005). The Poly1305-AES message-authentication code. In *International workshop on fast software encryption* (pp. 32-49). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11502760_3

Bichsel, P., Binding, C., Camenisch, J., Groß, T., Heydt-Benjamin, T., Sommer, D., & Zaverucha, G. (2009). *Cryptographic protocols of the identity mixer library.* Tech. Rep. RZ 3730, Tech. Rep. Retrieved from http://patrik.biche.ch/pub/rz3730.pdf

Chaum, D. (1985). Showing credentials without identification. In *Workshop on the Theory and Application of of Cryptographic Techniques* (pp. 241-244). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39805-8_28

Camenisch, J., Drijvers, M., Dzurenda, P., & Hajny, J. (2019). Fast keyed-verification anonymous credentials on standard smart cards. In *IFIP International Conference on ICT Systems Security and Privacy Protection* (pp. 286-298). Springer, Cham. https://doi.org/10.1007/978-3-030-22312-0_20

Camenisch, J., & Lysyanskaya, A. (2001). An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International conference on the theory and applications of cryptographic techniques* (pp. 93-118). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44987-6_7

Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G. L., Neven, G., & Pedersen, M. Ø. (2014). *Scientific comparison of ABC protocols.* Retrieved from https://abc4trust.eu/download/Deliverable D3.1 Part 1.pdf

Camenisch, J., & Stadler, M. (1997). Efficient group signature schemes for large groups. In *Annual International Cryptology Conference* (pp. 410-424). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0052252

Chase, M., Meiklejohn, S., & Zaverucha, G. (2014). Algebraic MACs and keyed-verification anonymous credentials. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1205-1216). https://doi.org/10.1145/2660267.2660328

Chase, M., Perrin, T., & Zaverucha, G. (2020). The signal private group system and anonymous credentials supporting efficient verifiable encryption. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1445-1459). https://doi.org/10.1145/3372297.3417887

Couteau, G., & Reichle, M. (2019l). Non-interactive keyed-verification anonymous credentials. In *IACR International Workshop on Public Key Cryptography* (pp. 66-96). Springer, Cham. https://doi.org/10.1007/978-3-030-17253-4_3

Dodis, Y., Kiltz, E., Pietrzak, K., & Wichs, D. (2012). Message authentication, revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 355-374). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29011-4_22

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory, 31*(4), 469-472. https://doi.org/10.1109/TIT.1985.1057074.

Fuchsbauer, G., Hanser, C., & Slamanig, D. (2019). Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology, 32*(2), 498-546. https://doi.org/10.1007/s00145-018-9281-4

Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof systems[J]. *SIAM Journal on computing*, *18*(1), 186-208. https://doi.org/10.1137/0218012

Kerry, C. F., & Director, C. R. (2013). *FIPS PUB 186-4 federal information processing standards publication digital signature standard (DSS).* Retrieved from https://cryptome.org/2013/07/NIST.FIPS.186-4.pdf

Krawczyk, H., Bellare, M., & Canetti, R. (1997). *HMAC: Keyed-hashing for message authentication.* Retrieved from https://www.hjp.at/doc/rfc/rfc2104.html

Paquin, C., & Zaverucha, G. (2011). *U-prove cryptographic specification v1.1*. Technical Report, Microsoft Corporation. Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/U-Prove20Cryptographic20Specification20V1.1.pdf

Sanders, O. (2020). Efficient Redactable Signature and Application to Anonymous Credentials. *Public Key Cryptography, 2*, 628-656. https://doi.org/10.1007/978-3-030-45388-6_22

Schnorr, C. P. (1991). Efficient signature generation by smart cards. *Journal of cryptology, 4*(3), 161-174. https://doi.org/10.1007/BF00196725

Tan, S. Y., & Groß, T. (2020). Monipoly—an expressive q-SDH-based anonymous attribute-based credential system. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 498-526). Springer, Cham. https://doi.org/10.1007/978-3-030-64840-4_17

Torr, C., & Mayes, K. (2017). Multos and multos application development. In *Smart Cards, Tokens, Security and Applications* (pp. 469-496). Springer, Cham. https://doi.org/10.1007/978-3-319-50500-8_17

Appendix A

Security Analysis of Our Keyed-Verification Anonymous Credentials

We now formally prove that our keyed-verfication anonymous credential (KVAC) system is secure.

Theorem 1 (Correctness). If the algebraic MAC scheme, the Elgamal encryption scheme, and the zero-knowledge proof are correct, our KVAC system is correct.

Proof. This follows directly from the completeness of the algebraic MAC scheme (Camenisch, Drijvers, Dzurenda & Hajny 2019), the Elgamal encryption scheme (ElGamal, 1985) and the completeness of the two zero knowledge proofs $\boldsymbol{\pi}_{MAC}$ described in the protocol (*CredObtain, CredIssue*) and $\boldsymbol{\pi}$ described in the protocol (*CredProve, CredVerify*). One can easily check the correctness of our KVAC system.

Theorem 2 (Unforgeability). If the algebraic MAC scheme is unforgeable, our KVAC system is unforgeable.

Proof. We require any attacker cannot forge cannot a credential on attributes that does not belong to him. In our KVAC system, a credential $\sum$ is an algebraic MAC (Camenisch, Drijvers, Dzurenda & Hajny 2019) on messages (uid, $\boldsymbol{M}$). Thus, if an attacker tries to forge a new credential $\sum^{*}$ on (uid^{*}, $\boldsymbol{M}^{*}$) that not signed by the MAC secret key, it will directly break the unforgeability of the algebraic MAC, which have been proven by Camenisch, Drijvers, Dzurenda & Hajny (2019).

Theorem 3 (Anonymity). Our KVAC system is anonymous in the random oracle model.

Proof. We constrcuct an algorithm *SimCredProve*(*par, **M**, D, tpk, uid*) to prove the anonymity of our KVAC system. Firstly, it encrypts his user identifier uid to get $nym = (nym_1, nym_2)$ as usual. Then, it selects a random integer $\sum' \in Z_p$ and simulates the corresponding zero knowledge proof by taking some random integers s_r, s_k, s_{uid}, and s_{m_i} for all $i \notin D$, and c, which all integers in Z_p. Afterwards, it computes t_1, t_2, and t_3 as follows:

$$t_1 = g^{s_r} \cdot (\sum')^{-c \cdot x + x_0 \cdot s_{uid} + \sum_{i \notin D}(x_i \cdot s_{m_i}) - \sum_{i \in D}(x_i \cdot m_i \cdot c)};$$

$$t_2 = (nym_1)^{c} \cdot g^{-s_k};$$

$$t_3 = g^{s_{uid}} \cdot (tpk)^{s_k} \cdot (nym_2)^{-c}.$$

Finally, it programs the random oracle to compute a challenge c as follows:

$$c = Hash(D, \boldsymbol{M}_D, t_1, t_2, t_3, \sigma', par, ipar, nym, nonce).$$

Obviously, the output of *SimCredProve*(*par, **M**, D, tpk, uid*) is identically distributed to the output generated by a real prover in the algorithm *CredProve*(*par,* $\sum$*, **M**, D, tpk, uid*).

Theorem 4 (Traceability). If the Elgamal encryption scheme is correct, our KVAC system is traceable.

Proof. In each run of the protocol (*CredProve, CredVerify*), a user needs to present a pseudonym $nym = (nym_1, nym_2) = (g^k, tpk^k\, g^{uid})$ which is obtained by encrypting his user identifier imformation g^{uid} and a zero-knowledge proof $\boldsymbol{\pi}$ which proves that the pseudonym nym is correctly computed using the tracing authority public key tpk. Consequently, when a verifier accepts the zero-knowledge proof $\boldsymbol{\pi}$, then the tracing authority is able to decrypt the nym, demonstrating the traceability of our KVAC system.

Theorem 5 (Unlinkability). Our KVAC system is unlinkable.

Proof. In each run of the protocol (*CredProve, CredVerify*), a user needs to select a random integer $r \in Z_p$ and randomize his credential $\sum = (\sigma, \sigma_{x_1}, \ldots, \sigma_{x_n})$ by computing:

$$\sum' = \sigma^{r} \text{ and } t_1 = \sigma_{x_0}{}^{\rho_{uid} \cdot r} \cdot \prod_{i \notin D} \sigma_{x_i}{}^{\rho_{m_i} \cdot r} \cdot g^{r}$$

Obviously, σ^{r} is identically distributed to the σ. Moreover, t_1 does not leak any information about ($\sigma_{x_1}, \ldots, \sigma_{x_n}$). Therefore, any verifier who is also the credential issuer cannot link the showing credential to its origin.