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

Multi-Issuer Attribute-based Anonymous Credential with Traceability and Revocation

Ye Yang[1]

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

Correspondence: Ye Yang, College of Cyber Security, Jinan University, Guangzhou, China.

Received: February 11, 2021 Accepted: March 16, 2022 Online Published: March 17, 2022

doi:10.5539/cis.v15n2p68 URL: https://doi.org/10.5539/cis.v15n2p68

Abstract

Attribute-based anonymous credential schemes allow users to obtain credentials from the issuer and prove the possession of their attributes interactively and anonymously with service providers. So far, most existing schemes only consider single-issuer, where some of them are extended to be traceable or revocable. In the reality, anonymous credential schemes under multi-issuer are more practical, since users could query for credentials from different issuers and use some of them simultaneously, which is more efficient than showing them individually. Although there are also multi-issuer schemes where users obtain credentials from different issuers and show an aggregated credential to service providers, however, these schemes lack practical properties, for example, revocation of invalid users. In this paper, we propose a multi-issuer attribute-based anonymous credential with traceability and revocation, which provides traceability of invalid users, and revocation of the specific users. Users receive credentials from multiple issuers and show an aggregated credential of selective disclosures of attributes. We provide the security model of our scheme of anonymity and unforgeability. Finally, we discuss the computational complexity, which shows the practicality and efficiency of our scheme.

Keywords: anonymous credential, multi-issuer, traceability, revocation

1. Introduction

Anonymous credential (AC) was first introduced to allow users to anonymously interact with issuers. Users obtain credentials from issuers and show their credentials to verifiers. Subsequently, attribute-based anonymous credential (ABC) was proposed, which allows users to obtain credentials of their attributes from issuers and prove the possession of these attributes to verifiers. More specifically, users could disclose some of their attributes or just prove some relations during showing their credentials. Commonly, ABC schemes consider one issuer and multiple users. Ideally, situations under multiple issuers are more practical. Although single-issuer ABC schemes can be extended to multi-authority, where the user obtains credentials from different issuers and separately shows each needed credential to verifiers, and the computation cost is linear to the number of credentials. Therefore, multi-authority ABC schemes are proposed to solve this problem. As shown in Figure 1, existing multi-authority ABC schemes use aggregatable signatures, and issue per attribute a credential. User aggregates the needed credentials and shows the aggregated credential, which leads to constant computation cost.

So far, various properties have been considered in ABC schemes, such as traceability, revocation, and delegation. Traceability is that tracing authority (TA) traces out the concrete user when some user invalidly uses credentials. Revocation, run by revocation authority (RA), occurs after TA traces out the concrete user or user calls for an active revocation after losing their secret key. Delegation allows delegatable users to issue a more restricted credential to others. There are many single-issuer ABC schemes that consider some of these properties. When it comes to multi-authority ABC schemes, only Hébant and Pointcheval (2020) consider traceability. In this paper, we propose a multi-authority ABC scheme with traceability and revocation, and we next show the intuition of our scheme.

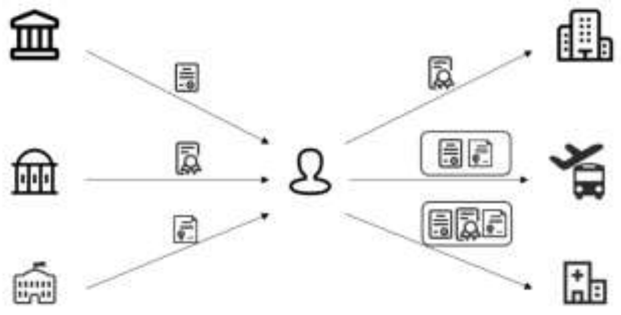

Figure 1. Multi-Issuer Attribute-based Anonymous Credential

1.1 Intuition

First, we review the aggregatable attribute-based equivalence class signature proposed by Hanzlik and Slamanig (2021). Assume that attributes can be described as an attribute name $Attr$ and an attribute value val, for example, ("age", 18) represents the age of 18. In this scheme, there is a main secret key msk issuing signing keys sk_{Attr} for each attribute name $Attr$. To sign a message vector $M \in (\mathbb{G}_i^*)^l (i \in \{1,2\})$ with an attribute $(Attr, val)$, the signer first calculates the randomness $y \leftarrow F(\kappa, M)$ using the pseudo-random function on inputting the PRF key κ. Then the signer computes:

$$Z_1 = \left(\prod_{i=1}^{\ell} M_i^{x_i}\right)^y, Y_1 = g^{1/y}, Y_2 = \hat{g}^{1/y}, V_2 = H_{Attr}(v_{Attr})^{1/y}$$

and parse $\sigma = (Z_1, Y_1, Y_2, V_2)$ as the signature, where $H_{Attr}(\cdot) = H(pk_{Attr} || \cdot)$. The intuition of this signature is to share the same randomness $y \leftarrow F(\kappa, M)$ among different attributes under the same message M. Therefore, n signatures $\{Z_{1,i}, Y_{1,i}, Y_{2,i}, V_{2,i}\}_{i\in[n]}$ under the same message M can be aggregated by aggregating $\{Z_{1,i}\}_{i\in[n]}$ and $\{V_{2,i}\}_{i\in[n]}$ respectively and use one of the $\{Y_{1,i}, Y_{2,i}\}_{i\in[n]}$.

Similarly, when constructing a multi-authority ABC scheme using this aggregatable attribute-based equivalence class signature, different signers represent multiple issuers and each user uses a unique message M. Different from the above scheme under single-authority using the same signature (Hanzlik & Slamanig, 2021), we need to solve the problem that different signers should share the same randomness y among different attributes under the same message M without revealing the randomness to users. Diffie-Hellman key exchange protocol allows two parties to share a secret key via public communications. This protocol can be extended to triple or even more parties. Therefore, using this protocol, all signers of the system agree on the same randomness for each user. User can aggregate any signatures as long as the received signatures are signed for him.

Generally, user owns a pair of secret key and public key (usk, g^{usk}), where $g \in \mathbb{G}_i (i \in \{1,2\})$. To realize revocation with accumulator, we add g_p^{usk} into user's public key, where $g_p \in \mathbb{Z}_p^*$ and usk is user's secret key, since that accumulated element must be $\mathbb{Z}_p^*$ in existing accumulator schemes. However, this method leads to a problem that users could generate multiple g_p^{usk} with random exponents except the real usk. Therefore, users need to prove the consistency of the usk used in g^{usk} and g_p^{usk}, and register their identities by calling RA to add g_p^{usk} into RA's accumulator. RA revokes one user by removing them from the accumulator and updating the accumulator value and membership witnesses for valid elements. Inspired by a traceable scheme (Blömer & Bobolz, 2018), we use the same method to realize traceability. Users encrypt their g^{usk} under TA's public key as their pseudonyms and give out the corresponding proofs of possession when using credentials. TA can trace out the concrete user by decrypting the pseudonym.

1.2 Contribution

We propose a multi-authority attribute-based anonymous credential scheme with traceability and revocation. Specifically, our contributions are as follows:

1) We construct the first multi-authority attribute-based anonymous credential with traceability and revocation, which can traces out the concrete user when some user breaks the law et al., and revoke the user.

2) We give out the security model and analyze the computational complexity, which shows the practicality and efficiency of our scheme.

1.3 Organization

In the rest of our paper, we introduce our related work and preliminaries respectively in Section 2 and Section 3. In Section 4, we present the syntax and security model of our scheme. Then we show the construction and security analysis in Section 5. The computational complexity analysis is shown in Section 6. Finally, we conclude our scheme in Section 7.

2. Related Work

Anonymous credential was first introduced by Chaum (1985) allowing users to interact with issuers anonymously. Then Brands (2000) proposed an anonymous credential scheme for single-show, and later multi-show anonymous credential scheme was proposed by Camenisch and Lysyanskaya (2001). Subsequently, strengthened anonymous credential schemes with different constructions are proposed.

There are also schemes targeting various properties. Traceable anonymous credential constructed using sanitizable signatures (Canard & Lescuyer, 2013) was found to be insecure by Hébant and Pointcheval (2020) that public elements would break the semantic security of ElGamal encryption as well as the anonymous credential scheme. Kaaniche and Laurent (2016) constructed a traceable anonymous credential with attribute-based signatures, which was also found to be insecure by Vergnaud (2017). Blömer and Bobolz (2018) proposed a delegatable and traceable anonymous credential, where TA uses Cramer-Shoup encryption (Cramer & Shoup, 1998) to decrypt ciphertext and obtains user's public key. Hébant and Pointcheval (2020) proposed traceable anonymous credential schemes under multi-authority using one-time aggregatable signatures with randomized tags and bounded aggregatable signatures. The tracing method is adding the extra user tracing key to user and testing if the bilinear pairing equations hold. Lapon et al. analyzed the revocation methods used in revocable anonymous credential (Lapon, Kohlweiss, Decker, & Naessens, 2011), such as accumulator (Acar, & Nguyen, 2011; Au, Tsang, Susilo, & Mu, 2009; Camenisch, Kohlweiss, & Soriente, 2009; Camenisch, & Lysyanskaya, 2002; Derler, Hanser, & Slamanig, 2015; Nguyen, 2005), verifier local revocation (Camenisch, Drijvers, & Hajny, 2016), revocation list. Typically, there is a pseudonym linked with the credential, thus, revocation of the credential is the revocation of the pseudonym. Since accumulator can be expressed as a value for a set of accumulator values and provides membership witnesses for each of them, it is widely used for revocation as a whitelist (Au et al., 2009; Camenisch et al., 2009; Camenisch, & Lysyanskaya, 2002; Nguyen, 2005) or blacklist(Acar, & Nguyen, 2011; Derler, Hanser, & Slamanig, 2015) putting the valid or invalid item into it.

3. Preliminaries

3.1 Bilinear Map

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p, e be an efficiently computable bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1) Bilinear: For all generators $g \in \mathbb{G}_1, \hat{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab}$.

2) Non-degenerate: For all generators $g \in \mathbb{G}_1^*, \hat{g} \in \mathbb{G}_2^*$, $e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$.

3) Efficient: For all generators $g \in \mathbb{G}_1, \hat{g} \in \mathbb{G}_2$, $e(g, \hat{g})$ is efficiently computable.

In this paper, we use Type-3 bilinear group (Galbraith, Paterson, & Smart, 2008), which means that $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficient computable homomorphism between them. In the rest of our paper, we parse $BG = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p) \leftarrow BGGen(1^\lambda)$, where λ is a security parameter.

3.2 Aggregatable Attribute-based Equivalence Class Signatures

Structure-preserving signatures on equivalence classes (SPS-EQ) (Hanser & Slamanig, 2014; Fuchsbauer, Hanser, & Slamanig, 2019) sign equivalence classes $[M]$ of vectors $M \in (\mathbb{G}_i^*)^l (i \in \{1,2\})$ of length l with equivalence relation: $M, M' \in \mathbb{G}_i^l: M \sim_R M' \Leftrightarrow \exists \rho \in \mathbb{Z}_p^*: M' = M^\rho$. Recently, Hanzlik and Slamanig (2021) propose an aggregatable attribute-based equivalence class signature (AAEQ). Assume that attributes can be described as an attribute name $Attr$ and an attribute value val, for example, ("age", 18) represents the age of 18. In such a scheme, the main secret key can issue signing keys for attributes ($Attr$). When signing a message M or a representative of a class $[M]$ with an attribute signing key, the signing process additionally takes the attribute value (val) as input. Moreover, signatures under different attribute signing keys can be aggregated into a valid one if and only if they are signing the same representative M of a class. AAEQ consists of the following algorithms:

- $AAEQ.Setup(1^\lambda, t, l) \to (msk, mpk)$: It takes security parameter λ, number of attribute types t, and length of signed message l as input. It first runs $BGGen(1^\lambda)$ to obtain $BG = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$, with generators $g \leftarrow \mathbb{G}_1, \hat{g} \leftarrow \mathbb{G}_2$. Parse $H: \{0,1\}^* \rightarrow \mathbb{G}_2$. Then it chooses PRF key $\kappa \leftarrow \mathcal{K}$ and for $i \in [t]$: choose $\vec{x}_i \leftarrow (\mathbb{Z}_p^*)^l$ and set $pk_{Attr_i} = (\hat{g}^{\vec{x}_i}), sk_{Attr_i} = (pk_{Attr_i}, \vec{x}_i, \kappa)$. It outputs $msk = (sk_{Attr_1}, \ldots, sk_{Attr_t}), mpk = (pk_{Attr_1}, \ldots, pk_{Attr_t})$.

• $AAEQ.AKGen(msk, Attr) \rightarrow sk_{Attr}$: It takes msk and $Attr \in [t]$ as input, and outputs $msk[Attr]$.

• $AAEQ.Sign(sk_{Attr}, v_{Attr}, M) \rightarrow \sigma$: Parse $sk_{Attr} = (pk_{Attr}, \vec{x}, k), v_{Attr} \in \{0,1\}^*, M \in (\mathbb{G}_1^*)^l$. It first computes $y \leftarrow F(\kappa, M)$. Set $H_{Attr}(\cdot) = H(pk_{Attr} || \cdot)$, compute $Z_1 = \left(\prod_{i=1}^{\ell} M_i^{x_i}\right)^y, Y_1 = g^{1/y}, Y_2 = \hat{g}^{1/y}, V_2 = H_{Attr}(v_{Attr})^{1/y}$. It outputs $\sigma = (Z_1, Y_1, Y_2, V_2)$.

• $AAEQ.ChgRep(M, \sigma, \mu, mpk) \rightarrow (M', \sigma')$: Parse $M \in (\mathbb{G}_1^*)^l, \mu \in \mathbb{Z}_p^*$. It first chooses $\psi \leftarrow \mathbb{Z}_p^*$, and compute $M' = M^\mu, \sigma' = (Z_1^{\psi\mu}, Y_1^{1/\psi}, Y_2^{1/\psi}, V_2^{1/\psi})$.

• $AAEQ.Agg(mpk, \{\sigma_i\}) \rightarrow \sigma'$: Given mpk and set of valid signatures $\{\sigma_i\} = \{Z_{1,i}, Y_{1,i}, Y_{2,i}, V_{2,i}\}_i$, it returns $\perp$ if $Y_{1,i} \neq Y_{1,j}$ or $Y_{2,i} \neq Y_{2,j}$ for $i \neq j$. Otherwise, it returns $\sigma' = (\prod_{i=1}^{k} Z_{1,i}, Y_{1,1}, Y_{2,1}, \prod_{i=1}^{k} V_{2,i})$.

• $AAEQ.Verify(mpk, A, \sigma', M) \rightarrow 1/0$: Parse $mpk = (pk_{Attr_1}, \ldots, pk_{Attr_t}), A = (Attr_i, v_{Attr_i})_{i \in [t]} \in ([t], \{0,1\}^*), \sigma = (Z_1, Y_1, Y_2, V_2), M \in (\mathbb{G}_1^*)^l$. If the following equations hold, it returns 1, otherwise returns 0: $\prod_{i=1}^{\ell} e(M_i, \prod_{j=1}^{t} pk_{Attr_j,i}) = e(Z_1, Y_2) \wedge e(Y_1, \hat{g}) = e(g, Y_2) \wedge e\left(Y_1, \prod_{j=1}^{t} H_{Attr_j}\left(v_{Attr_j}\right)\right) = e(g, V_2)$.

3.3 Dynamic Accumulator

A cryptographic accumulator allows a set of elements to be accumulated into a single value and gives membership witness for them indicating that they are in the accumulated set. An accumulator is dynamic if elements can be added into or deleted from the accumulator set, and the corresponding accumulator value and membership can be updated. In this paper, we use the dynamic accumulator (Au et al., 2009) as a whitelist as follows:

• $ACC.Gen(BG, n) \rightarrow (sk_\Pi, pk_\Pi)$: Given $BG = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$ and maximum accumulated number n, it first chooses generators $g \leftarrow \mathbb{G}_1, \hat{g} \leftarrow \mathbb{G}_2$. Then it randomly chooses $\alpha \leftarrow \mathbb{Z}_p^*$ and computes: $(g_i)_{i \in [n]} = \left(g^{\alpha^i}\right)_{i \in [n]}, \left(\hat{g}_i\right)_{i \in [n]} = \left(\hat{g}^{\alpha^i}\right)_{i \in [n]}$. It returns $sk_\Pi = \alpha, pk_\Pi = (BG, g, \hat{g}, (g_i)_{i \in [n]}, (\hat{g}_i)_{i \in [n]})$.

• $ACC.Eval(\mathcal{X}, sk_\Pi, pk_\Pi) \rightarrow (\Pi_\mathcal{X}, aux)$: Parse $\mathcal{X} = \{x_1, \ldots, x_n\} \in \mathbb{Z}_p^*$ be n accumulated elements. Compute $\pi(X) = \prod_{i \in [n]}(x_i + \alpha) = \sum_{i=0}^{n} u_i \cdot \alpha^i$, set $\Pi_\mathcal{X} = g_0^{\pi(X)} = \prod_{i=0}^{n} g_i^{u_i} \in \mathbb{G}_1$. It returns $\Pi_\mathcal{X}$ as the accumulator value, and parses $aux = \mathcal{X} = \{x_1, \ldots, x_n\}$ as the auxiliary infomation.

• $ACC.Create(\Pi_\mathcal{X}, aux, y, sk_\Pi, pk_\Pi) \rightarrow \omega_y$: It takes the accumulator value $\Pi_\mathcal{X}$, auxiliary infomation aux, the accumulated element y, secret key sk_Π, and public key pk_Π as input. If $y \in \mathcal{X}$, it computes $\pi(X) = \prod_{i \in [n]}(x_i + \alpha) = \sum_{i=0}^{n} u_i \cdot \alpha^i$ and $g(X) = \prod_{i=1.x_i \neq y}^{n}(x_i + \alpha) = \sum_{i=0}^{n-1} u_i \cdot \alpha^i$, where $\pi(X) = g(X)(y + \alpha)$. Then it returns $\omega_y = \prod_{i=0}^{n-1} \hat{g}_i^{u_i}$ as the membership witness for y.

• $ACC.Verify(\Pi_\mathcal{X}, \omega_y, y, pk_\Pi) \rightarrow 1/0$: It takes the accumulator value $\Pi_\mathcal{X}$, membership witness ω_y, the accumulated element y, and public key pk_Π as input. If $e(\Pi_\mathcal{X}, \hat{g}_0) = e(g_0^y g_0^\alpha, \omega_y)$, it returns 1. Otherwise, it returns 0.

• $ACC.AccUpdate(\Pi_\mathcal{X}, y', sk_\Pi, pk_\Pi) \rightarrow \Pi'_\mathcal{X}$: It takes the accumulator value $\Pi_\mathcal{X}$, the element y', secret key sk_Π, and public key pk_Π as input. If y' is added into the accumulator, it updates the accumulator value $\Pi'_\mathcal{X} = \Pi_\mathcal{X}^{y'+\alpha}$. If y' is deleted from the accumulator, it updates the accumulator value $\Pi'_\mathcal{X} = \Pi_\mathcal{X}^{1/(y'+\alpha)}$.

• $ACC.WitUpdate\left(\Pi'_\mathcal{X}, \omega_y, y', sk_\Pi, pk_\Pi\right) \rightarrow \omega'$: It takes the updated accumulator value $\Pi'_\mathcal{X}$, membership witness ω_y for y, the element y', secret key sk_Π, and public key pk_Π as input. If y' is added into the accumulator, it updates the membership witness for y: $\omega'_y = \omega_y^{y'+\alpha}$. If y' is deleted from the accumulator, it updates the membership witness for y: $\omega'_y = \omega_y^{1/(y'+\alpha)}$.

3.4 Cramer-Shoup Encryption

Cramer-Shoup encryption system (Cramer & Shoup, 1998) is an asymmetric key encryption system that was first proven to be secure against chosen ciphertext attack. The scheme consists of the following algorithms:

• $CS.Setup(1^\lambda) \rightarrow pp_{CS}$: Given the security parameter λ, this algorithm first chooses a cyclic group G with prime order q, and a collision-resistant hash function: $H: \{0,1\}^* \rightarrow G$. It outputs $pp_{CS} = (G, q, H)$.

• $CS.KeyGen(pp_{CS}) \rightarrow (sk_{CS}, pk_{CS})$: Given $pp_{CS} = (G, q, H)$, this algorithm first randomly chooses generators $g_1, g_2 \leftarrow G$ and $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 \leftarrow \mathbb{Z}_q$. Then it computes $A = g_1^{\alpha_1} g_2^{\alpha_2}, B = g_1^{\beta_1} g_2^{\beta_2}, C = g_1^{\gamma_1} g_2^{\gamma_2}$. It outputs $sk_{CS} = (\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2), pk_{CS} = (g_1, g_2, A, B, C)$.

• $CS.Encrypt(pp_{CS}, pk_{CS}, m) \rightarrow c$: Given $pp_{CS} = (G, q, H)$, $pk_{CS} = (g_1, g_2, A, B, C)$ and message $m \in G$, this algorithm first randomly chooses $r \leftarrow \mathbb{Z}_q$ and computes: $c_1 = g_1^r, c_2 = g_2^r, c_3 = C^r \cdot m, c_4 = A^r B^{wr}$, where $w = H(c_1, c_2, c_3)$. It outputs ciphertext $c = (c_1, c_2, c_3, c_4)$.

• $CS.Decrypt(pp_{CS}, sk_{CS}, c) \to m/\perp$: Given $pp_{CS} = (G, q, H), sk_{CS} = (\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2)$, and ciphertext $c = (c_1, c_2, c_3, c_4)$, this algoritem first computes $w = H(c_1, c_2, c_3)$ and checks whether $c_4 = c_1^{\alpha_1 + w\beta_1} c_2^{\alpha_2 + w\beta_2}$ holds. If the equation does not hold, it returns $\perp$. Otherwise, it computes and outputs $m = c_3 \cdot c_1^{-\gamma_1} \cdot c_2^{-\gamma_2}$.

3.5 Diffie-Hellman Key Exchange

Diffie-Hellman key exchange is one of the first cryptographic key exchanging methods over insecure public channels, which was conceived by Ralph Merkle (1978) and named after Whitfield Diffie and Martin Hellman (Diffie, & Hellman, 1982). It is used to share a secret key between two parties via public communications. Denote G is a cyclic group with prime order, g is a generator of G. Assume that Alice and Bob want to agree on a key for communication. Steps are as follows:

1) Alice randomly chooses $a \leftarrow \mathbb{Z}_p^*$ and computes g^a to Bob.

2) Bob randomly chooses $b \leftarrow \mathbb{Z}_p^*$ and computes g^b to Alice.

3) Alice computes $(g^b)^a = g^{ab}$, Bob computes $(g^a)^b = g^{ab}$.

Since Alice and Bob both possess g^{ab}, it can be seen as the secret key for their communication. Obviously, the key exchange protocol is not limited to two parties. It can be extended to triple, or more parties, as long as every two participants run the above steps and agree on the final key (Kudla, & Paterson, 2005). Although this method is insecure against man-in-the-middle attack, variants of Diffie-Hellman key exchange protocols are proposed to avoid these attacks, for example, Station-to-Station protocol (Diffie, Van, & Wiener, 1992).

3.6 Proof of Knowledge

In this paper, we use the zero-knowledge proof (Goldwasser, Micali, & Rackoff, 1989) to prove knowledge of discrete logarithms, and relations between them. Denote $POK[(w); (x, w) \in R]$ as zero-knowledge proof of knowledge, where the prover proves knowledge of w such that (x, w) satisfies some NP relations R without revealing any information about w. A zero-knowledge proof has three properties:

• Completeness: If the prover provides a true statement, the verifier always accepts it, except with negligible probability.

• Soundness: If the prover cheats and provides a false statement, the verifier always rejects it, except with negligible probability.

• Zero-knowledge: If the prover's statement is true, the verifier learns nothing except for the fact that statement is true. For any verifier, there exists a polynomial-time simulator that produces transcripts indistinguishable from the view what the verifier produces with the same input.

4. Attribute-based Anonymous Credential

In this paper, we propose a multi-authority attribute-based anonymous credential. There exist multiple authorities issuing credentials for users. After users obtain their credentials for each attribute, they can aggregate the necessary credentials and prove their possession of them to service providers. When some users do something invalid or expire in the system, the tracing authority (TA) can find out the concrete users. After TA traces out users, or when users initiatively call for revocation after losing their secret key, the revocation authority (RA) can revoke it. Additionally, users should prove the validity of their identities when showing their credentials.

4.1 Syntax

Our scheme consists of the following algorithms:

• $Setup(1^\lambda, K) \to (pp)$: This algorithm takes the security parameter λ and number of issuers as input, and outputs the public parameters pp. Issuers in the system agree on these public parameters.

• $IssuerKeyGen(pp, t, l) \to (isk, ipk)$: This algorithm run by each issuer takes the public parameters pp, number of attribute types t, and length of signed message l as input, and outputs issuer's public key ipk and secret key isk.

• $TAKeyGen(pp) \to (tsk, tpk)$: This algorithm run by TA takes the public parameters pp as input, and outputs the tracing secret key tsk and public key tpk.

• $RAKeyGen(pp, n) \to (rsk, rpk)$: This algorithm run by RA takes the public parameters pp and number of maximum accumulated elements n as input, and outputs the revocation secret key rsk and public key rpk.

• $UKeyGen(pp) \rightarrow (usk, upk)$: This algorithm run by each user takes the public parameters pp as input, and outputs user's secret key usk and public key upk.

• $NymGen(pp, tpk, upk) \rightarrow (nym, sk_u)$: This algorithm run by each user takes the public parameters pp, TA's public key and user's secret key usk as input, and outputs user's pseudonym nym and the corresponding secret sk_u.

• $Trace(pp, tsk, nym) \rightarrow upk$: This algorithm run by TA takes the public parameters pp, tracing secret key tsk and user's pseudonym nym as input, and outputs the concrete user's public key.

• $Revoke(pp, rsk, rpk, upk) \rightarrow \mathbb{R}$: This algorithm run by RA takes the public parameters pp, revocation secret key rsk, revocation public key rpk, and user's public key upk as input, and outputs the revocation information $\mathbb{R}$.

• $Issue(pp, ipk, isk, \vec{A}, upk) \leftrightarrow Obatin(pp, \vec{A}, upk, usk, \omega) \rightarrow cred$: $Obatin$ run by the user takes the public parameters pp, user's attributes $\vec{A}$, user's public key upk, secret key usk, and membership witness of user ω as input. $Issue$ run by the issuer takes the public parameters pp, issuer's public key ipk, issuer's secret key isk, user's attributes $\vec{A}$, and user's public key upk as input. These algorithms are run by issuers and users interactively, and finally outputs credentials.

• $Show(pp, \{ipk_k\}, tpk, rpk, upk, usk, \{\sigma_j\}, \vec{A}', \omega, \mathbb{R}) \leftrightarrow Verify(pp, avk, tpk, rpk, nym, \vec{A}', \sigma', \mathbb{R}) \rightarrow 0/1$: $Show$ run by the user takes the public parameters pp, k issuers' public key $\{ipk_k\}$, TA's public key tpk, RA's public key rpk, user's public key upk, user's secret key usk, credentials $\{\sigma_j\}$, user's disclosed attributes $\vec{A}'$, membership witness of user ω, and RA's revocation information $\mathbb{R}$ as input. $Verify$ run by the verifier takes the public parameters pp, aggregated verification key avk, TA's public key tpk, RA's public key rpk, user's pseudonym nym, user's disclosed attributes $\vec{A}'$, user's credential σ', and RA's revocation information $\mathbb{R}$ as input. These algorithms are run by users and verifiers interactively, and finally outputs 1 for success or 0 for failure.

4.2 Security Model

In this section, we first discuss the correctness of our scheme, then we give out the security model of our scheme from the aspect of anonymity and unforgeability. Anonymity is that credentials are unlinkable even when showing the same credential multiple times. Unforgeablity means that any malicious adversaries can not impersonate any honest users or forge any credentials not belonging to them for valid use.

Definition 4.1 (Correctness). A multi-authority attribute-based anonymous credential with traceability and revocability is correct if the following requirements hold:

• For all (usk, upk) generated by $UKeyGen(pp) \rightarrow (usk, upk)$ and any valid pseudonyms nym generated by $NymGen(pp, tpk, upk) \rightarrow (nym, sk_u)$, pseudonyms must be ciphertexts under TA's public key. TA can trace any users through their pseudonyms by $Trace(pp, tsk, nym) \rightarrow upk$.

• If issuers' public key ipk secret key isk, and user's public key are valid, credentials generated by $Issue(pp, ipk, isk, \vec{A}, upk) \leftrightarrow Obatin(pp, \vec{A}, upk, usk, \omega) \rightarrow cred$ are correct.

• If user's credential and pseudonym are valid, and zero-knowledge proofs are correct, user can successfully pass the verification in $Show(pp, \{ipk_k\}, upk, usk, \{\sigma_j\}, \vec{A}', \omega) \leftrightarrow Verify(pp, avk, nym, \vec{A}', \sigma') \rightarrow 0/1$.

• For all unrevoked users whose pseudonyms are in the accumulator, they can successfully obtain or show their credentials. For all revoked users whose pseudonyms are not in the accumulator, they cannot receive any new credentials from issuers or show any existed credentials to verifiers.

Definition 4.2 (Anonymity). A multi-authority attribute-based anonymous credential with traceability and revocability is anonymous if credentials are unlinkable for multiple uses.

Definition 4.3 (Unforgeability). A multi-authority attribute-based anonymous credential with traceability and revocability is unforgeable if malicious adversaries can not impersonate honest users or forge any credentials for valid use.

5. Construction

5.1 Our Scheme

In this section, we will give out our concrete construction. As mentioned in Section 1.1, in order to aggregate signatures from different issuers, issuers should use the same randomness y for the same user. Therefore, each time the user registers, all issuers agree on a randomness for this user using Diffie-Hellman key exchange protocol.

• $Setup(1^{\lambda}, K) \rightarrow (pp)$: Given the security parameter λ and the number of issuers, the system first runs $BGGen(1^{\lambda})$ and $CS.Setup(1^{\lambda})$ to obtain $BG = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p)$ and pp_{CS}, with generators $g \leftarrow \mathbb{G}_1, \hat{g} \leftarrow \mathbb{G}_2$. Denote $H: \{0,1\}^* \rightarrow \mathbb{G}_2, g_p \leftarrow \mathbb{Z}_p^*$. It outputs the public parameters $pp = (BG, pp_{CS}, g, \hat{g}, g_p, H)$. K Issuers in the system agree on these public parameters.

• $IssuerKeyGen(pp, t, l) \rightarrow (isk, ipk)$: Given the public parameters pp, number of attribute types t, and length of signed message l, each issuer j chooses a key κ for pseudonym random function F and $\vec{x}_i \leftarrow (\mathbb{Z}_p^*)^l$ and sets $ipk_j = pk_{Attr_i} = (\hat{g}^{\vec{x}_i}), isk_j = sk_{Attr_i} = (pk_{Attr_i}, \vec{x}_i, \kappa)$ for $i \in [t]$.

• $TAKeyGen(pp) \rightarrow (tsk, tpk)$: Given the public parameters pp, TA runs $CS.KeyGen(pp_{CS})$ to obtain sk_{CS} and pk_{CS}. Then it parses $tsk = sk_{CS}$ and outputs $tpk = pk_{CS}$.

• $RAKeyGen(pp, n) \rightarrow (rsk, rpk)$: Given the public parameters pp and number of maximum accumulated elements n, RA runs $ACC.Gen(BG, n)$ to obtain sk_{Π}, pk_{Π}. It parses $rsk = sk_{\Pi}$, and outputs $rpk = pk_{\Pi}$.

• $UKeyGen(pp) \rightarrow (usk, upk)$: Given the public parameters pp, the user randomly chooses $usk \leftarrow \mathbb{Z}_p^*$ and computes $upk = (upk_1, upk_2) = (g^{usk}, g_p^{usk})$. User gives upk to RA, along with proof of knowledge of the possession of usk and upk_1, upk_2 both use the same exponent: $POK[(usk); upk_1 = g^{usk} \wedge upk_2 = g_p^{usk}]$. Then RA adds upk_2 into list U and registers user: $ACC.Eval(U, rsk, rpk) \rightarrow (\Pi, aux)$, where U represents lists of valid users' public key, and generates membership witness for him: $ACC.Create(\Pi,\ aux,\ upk_2, rsk, rpk) \rightarrow \omega$. Issuers agree on a randomness for this user using Diffie-Hellman key exchanging protocol: 1) Issuer i computes $y_i = F(\kappa_i, upk_1)$ and gives $g_q^{y_i}$ to the other $K-1$ issuers. 2) After receiving elements from the other issuers, issuer i computes $y = g_q^{y_1 \cdots y_K}$. Thus, issuers use the same randomness for the same user.

• $NymGen(pp, tpk, upk) \rightarrow (nym, sk_u)$: Given the public parameters pp, TA's public key tpk and user's public key $upk = (upk_1, upk_2)$, user encrypts upk_2 using TA's tpk to obtain his pseudonym: $CS.Encrypt(pp_{CS}, tpk, upk_2) \rightarrow nym$. The randomness r used in the encryption is sk_u.

• $Trace(pp, tsk, nym) \rightarrow upk$: Given the public parameters pp and user's pseudonym, TA uses tsk to decrypt $CS.Decrypt(pp_{CS}, tsk, nym)$ and obtain user's public key upk_2.

• $Revoke(pp, rsk, rpk, upk) \rightarrow \mathbb{R}$: Given the public parameters pp and user's public key $upk = (upk_1, upk_2)$, RA first updates the list of valid users U by removing the upk_2 from the list. Then RA uses his secret key rsk and public key rpk to update the accumulator value $ACC.AccUpdate(\Pi, upk_2, rsk, rpk) \rightarrow \Pi'$ and membership witnesses for other valid users i : $ACC.WitUpdate(\Pi', \omega_i, upk_2, tsk, tpk) \rightarrow \omega_i'$. Membership witnesses are stored in $Wit[upk_{i,2}] = \omega_i'$. The algorithms output revocation information $\mathbb{R} = (\Pi', Wit)$.

• $Issue(pp, ipk, isk, \vec{A}, upk) \leftrightarrow Obatin(pp, \vec{A}, upk,\ usk, \omega) \rightarrow cred$: These algorithms run by the user and issuer interactively contain the following steps:

1) User gives his public key upk and attributes $\vec{A} = \{(Attr_1, val_1), \ldots, (Attr_{t'}, val_{t'})\}$, along with zero-knowledge proof of knowledge of possession of usk and witness ω to issuer i , where $t' \leq t$:

$$POK[(usk, \omega); upk_1 = g^{usk} \wedge upk_2 = g_p^{usk} \wedge ACC.Verify(\Pi, \omega, upk_2, rpk) = 1].$$

2) Issuer i checks the POK with user. If it does not hold, then the algorithm aborts. Otherwise, issuer computes signatures for each $Attr_j$ and gives $\{\sigma_{Attr_j}\}_{j \in [t']}$ to user as his credentials:

$$AAEQ.Sign(isk_i[Attr_j], val_j, upk_1) \rightarrow \sigma_{Attr_j}.$$

• $Show(pp,\ \{ipk_k\}, tpk, rpk, upk, usk, \{\sigma_j\}, \vec{A}', \omega, \mathbb{R}) \leftrightarrow Verify(pp, avk, tpk, rpk, nym, \vec{A}', \sigma', \mathbb{R}) \rightarrow 0/1$: These algorithms run by the user and service provider interactively contain the following steps:

1) User aggregates credentials of disclosed attributes $\vec{A}'$: $AAEQ.Agg(\{\sigma_j\}) \rightarrow \sigma$, and aggregates the corresponding issuers' public key: $avk = \{ipk_k\}$. Then he randomizes the signature and public key: $AAEQ.ChgRep(upk_1, \sigma, \mu, avk) \rightarrow (upk_1', \sigma')$ using the randomness μ. User generates pseudonym for himself and the credential: $NymGen(pp, tpk, upk) \rightarrow (nym, r_u)$. User calculates $R = g^{\mu}$ and gives upk_1', σ', nym, $\vec{A}'$, avk, R along with zero-knowledge proof of knowledge of possession of the credential:

$$POK[(usk, \mu, r_u, \omega, upk_2); R = g^{\mu} \wedge upk_1' = g^{\mu \cdot usk} \wedge\ nym = CS.Encrypt(pp_{CS}, tpk, upk_2; r_u) \wedge ACC.Verify(\Pi, \omega, upk_2, rpk) = 1].$$

2) Service Provider first checks the validity of the given signature: $AAEQ.Verify(avk, \vec{A}', \sigma', upk_1')$. If it does

not hold, the algorithm aborts. Otherwise, he checks the POK interactively with user.

5.2 Security Analysis

Theorem 5.1 (Correctness). If aggregatable attribute-based equivalence class signature, cramer-shoup encryption, dynamic universal accumulator, Diffie-Hellman key exchange protocol, and the zero-knowledge proofs are correct, our multi-issuer attribute-based anonymous credential scheme is correct.

Theorem 5.2 (Anonymity). If aggregatable attribute-based equivalence class signature provides perfect adaption, cramer-shoup encryption is CCA-secure, our multi-issuer attribute-based anonymous credential scheme is anonymous.

Theorem 5.3 (Unforeability). If aggregatable attribute-based equivalence class signature is unforgeable, cramer-shoup encryption is CCA-secure, dynamic universal accumulator is collision-resistant, our multi-issuer attribute-based anonymous credential scheme is unforgeable.

We omit details since the proofs are simple.

6. Computational Complexity

In this section, we analyze the computational complexity during issuing and showing a credential. In issuing process, each issuer performs 3 exponentiations in $\mathbb{G}_1$, 2 exponentiations in $\mathbb{G}_2$ to sign per attribute, plus 10 exponentiations in $\mathbb{G}_1$, 5 multiplications in $\mathbb{G}_1$, 5 pairings, 4 exponentiations in $\mathbb{G}_T$, and 4 multiplications in $\mathbb{G}_T$ for the interactive Schnorr-like proof with knowledge of usk, ω, upk_2 (Schnorr, 1991; Boneh, Boyen, & Shacham, 2004). User performs 7 exponentiations in $\mathbb{G}_1$, 1 exponentiation in $\mathbb{G}_2$, 2 multiplications in $\mathbb{G}_1$, 1 multiplication in $\mathbb{G}_2$, 3 pairings, 3 exponentiations in $\mathbb{G}_T$, and 2 multiplications in $\mathbb{G}_T$ for proof of possession of usk, ω, upk_2. In showing process, assume that user shows k attributes obtained from K issuers. The user performs k - 1 multiplications in $\mathbb{G}_1$, k - 1 multiplications in $\mathbb{G}_2$ to aggregate k signatures, and 3 exponentiations in $\mathbb{G}_1$, 2 exponentiations in $\mathbb{G}_2$ to change the aggregated signature into another one on equivalence classes, plus 7 exponentiations in $\mathbb{G}_1$, 1 exponentiation in $\mathbb{G}_2$, 2 multiplications in $\mathbb{G}_1$, 1 multiplication in $\mathbb{G}_2$, 3 pairings, 3 exponentiations in $\mathbb{G}_T$, 2 multiplications in $\mathbb{G}_T$ for proof of possession of $usk, \mu, r_u, \omega, upk_2$ (Schnorr, 1991; Boneh et al., 2004; Au, Susilo, & Mu, 2010). The verifier first performs 5 + k pairings and 2k multiplications in $\mathbb{G}_2$ to check the validity of the given signature. Then he performs 10 exponentiations in $\mathbb{G}_1$, 5 multiplications in $\mathbb{G}_1$, 5 pairings, 4 exponentiations in $\mathbb{G}_T$, 4 multiplications in $\mathbb{G}_T$ for the interactive proof of knowledge with user.

7. Conclusion

In this paper, we propose the first multi-issuer attribute-based anonymous credential scheme supporting traceability and revocation. We also give out security model and computational complexity analysis of our scheme, which indicates that our scheme is practical and efficient. For future work, we envisage extending our scheme with the delegation and revocation of credentials.

References

Acar, T., & Nguyen, L. (2011, March). Revocation for delegatable anonymous credentials. In *International Workshop on Public Key Cryptography* (pp. 423-440). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-19379-8_26

Au, M. H., Susilo, W., & Mu, Y. (2010, July). Proof-of-knowledge of representation of committed value and its applications. In *Australasian Conference on Information Security and Privacy* (pp. 352-369). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14081-5_22

Au, M. H., Tsang, P. P., Susilo, W., & Mu, Y. (2009, April). Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In *Cryptographers' track at the RSA conference* (pp. 295-308). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00862-7_20

Blömer, J., & Bobolz, J. (2018, July). Delegatable attribute-based anonymous credentials from dynamically malleable signatures. In *International Conference on Applied Cryptography and Network Security* (pp. 221-239). Springer, Cham. https://doi.org/10.1007/978-3-319-93387-0_12

Boneh, D., Boyen, X., & Shacham, H. (2004, August). Short group signatures. In *Annual international cryptology conference* (pp. 41-55). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-28628-8_3

Brands, S. (2000). Rethinking public key infrastructure and digital credentials. In *Rethinking Public Key Infrastructures Digital Certificates: Building in Privacy*. Cambridge, MA, USA: Mit Press.

https://doi.org/10.7551/mitpress/5931.001.0001

Camenisch, J., Drijvers, M., & Hajny, J. (2016, October). Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society* (pp. 123-133). https://doi.org/10.1145/2994620.2994625

Camenisch, J., Kohlweiss, M., & Soriente, C. (2009, March). An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *International workshop on public key cryptography* (pp. 481-500). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00468-1_27

Camenisch, J., & Lysyanskaya, A. (2001, May). 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., & Lysyanskaya, A. (2002, August). Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Annual international cryptology conference* (pp. 61-76). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45708-9_5

Canard, S., & Lescuyer, R. (2013, May). Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security* (pp. 381-392). https://doi.org/10.1145/2484313.2484363

Chaum, D. (1985). Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, *28*(10), 1030-1044. https://doi.org/10.1145/4372.4373

Cramer, R., & Shoup, V. (1998, August). A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Annual international cryptology conference* (pp. 13-25). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0055717

Derler, D., Hanser, C., & Slamanig, D. (2015, December). A new approach to efficient revocable attribute-based anonymous credentials. In *IMA International Conference on Cryptography and Coding* (pp. 57-74). Springer, Cham. https://doi.org/10.1007/978-3-319-27239-9_4

Diffie, W., & Hellman, M. E. (1982). New directions in cryptography. In *Secure communications and asymmetric cryptosystems* (pp. 143-180). Routledge.

Diffie, W., Van Oorschot, P. C., & Wiener, M. J. (1992). Authentication and authenticated key exchanges. *Designs, Codes and cryptography, 2*(2), 107-125. https://doi.org/10.1007/BF00124891

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

Galbraith, S. D., Paterson, K. G., & Smart, N. P. (2008). Pairings for cryptographers. *Discrete Applied Mathematics, 156*(16), 3113-3121. https://doi.org/10.1016/j.dam.2007.12.010

Goldwasser, S., Micali, S., & Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on computing, 18*(1), 186-208. https://doi.org/10.1137/0218012

Hanser, C., & Slamanig, D. (2014, December). Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 491-511). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-45611-8_26

Hanzlik, L., & Slamanig, D. (2021, November). With a little help from my friends: Constructing practical anonymous credentials. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 2004-2023). https://doi.org/10.1145/3460120.3484582

Hébant, C., & Pointcheval, D. (2020). Traceable constant-size multi-authority credentials. *Cryptology ePrint Archive*.

Kaaniche, N., & Laurent, M. (2016, September). Attribute-based signatures for supporting anonymous certification. In *European symposium on research in computer security* (pp. 279-300). Springer, Cham. https://doi.org/10.1007/978-3-319-45744-4_14

Kudla, C., & Paterson, K. G. (2005, December). Modular security proofs for key agreement protocols. In *International conference on the theory and application of cryptology and information security* (pp.

549-565). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11593447_30

Lapon, J., Kohlweiss, M., Decker, B. D., & Naessens, V. (2011, October). Analysis of revocation strategies for anonymous idemix credentials. In *IFIP International Conference on Communications and Multimedia Security* (pp. 3-17). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24712-5_1

Merkle, R. C. (1978). Secure communications over insecure channels. *Communications of the ACM, 21*(4), 294-299. https://doi.org/10.1145/359460.359473

Nguyen, L. (2005, February). Accumulators from bilinear pairings and applications. In *Cryptographers' track at the RSA conference* (pp. 275-292). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30574-3_19

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

Vergnaud, D. (2017). Comment on 'Attribute-Based Signatures for Supporting Anonymous Certification'by N. Kaaniche and M. Laurent (ESORICS 2016). *The Computer Journal, 60*(12), 1801-1808. https://doi.org/10.1093/comjnl/bxx058