Published in IEEE Transactions on Services Computing, 2023, 16 (5). DOI: 10.1109/TSC.2023.3280914

Threshold Attribute-Based Credentials With Redactable Signature

Rui Shi, Huamin Feng, Yang Yang, *Member, IEEE*, Feng Yuan, Yingjiu Li, Hwee Hwa Pang, and Robert H. Deng, *Fellow, IEEE*

Abstract—**Threshold attribute-based credentials are suitable for decentralized systems such as blockchains as such systems generally assume that authenticity, confidentiality, and availability can still be guaranteed in the presence of a threshold number of dishonest or faulty nodes. Coconut (NDSS'19) was the first selective disclosure attribute-based credentials scheme supporting threshold issuance. However, it does not support threshold tracing of user identities and threshold revocation of user credentials, which is desired for internal governance such as identity management, data auditing, and accountability. The communication and computation complexities of Coconut for verifying credentials are linear in the number of each user's attributes and thus costly. Addressing these issues, we propose a novel efficient threshold attribute-based anonymous credential scheme. While retaining all the features of Coconut, our scheme supports threshold tracing of user identities and threshold revocation of user credentials, and it significantly reduces the computational and communication complexities of credential verification. In addition, we prove that our scheme enjoys strong security features, including anonymity, blindness, traceability, and non-frameability.**

Index Terms—**Blockchain, threshold, traceable, unlinkable redactable signature, attribute-based credentials.**

Manuscript received 1 June 2022; revised 11 April 2023; accepted 22 May 2023. Date of publication 29 May 2023; date of current version 8 October 2023. The work of Yang Yang was supported in part by the National Natural Science Foundation of China under Grant 61872091, in part by Singapore National Research Foundation under Grant NRF2018NCR-NSOE004-0001, and by the AXA Research Fund. The work of Huamin Feng was supported by the National Defense Basic Research Program of China under Grant JCKY2019102C001. The work of Yingjiu Li was supported by Ripple University Blockchain Research Initiative. The work of HweeHwa Pang was supported by Lee Kong Chian Chair Professorship. The work of Robert Deng was supported in part by Singapore National Research Foundation under Grant NRF2018NCR-NSOE004-0001, and in part by AXA Research Fund. Recommended for acceptance by S. Nepal. *(Corresponding author: Yang Yang.)*

Rui Shi is with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China, and also with the Institute of Information Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China (e-mail: ruishi_mail@126.com).

Huamin Feng is with the Institute of Information Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China (e-mail: fenghm@besti.edu.cn).

Yang Yang is with the College of Computer Science and Big Data, Fuzhou University, Fuzhou 350116, China, and also with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: yang.yang.research@gmail.com).

Feng Yuan is with the Institute 706, Second Academy of CASIC, Beijing 100854, China (e-mail: fyuan1234@aliyun.com).

Yingjiu Li is with the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403 USA (e-mail: yingjiul@uoregon.edu).

Hwee Hwa Pang and Robert H. Deng are with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: hhpang@smu.edu.sg; robertdeng@smu.edu.sg).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSC.2023.3280914, provided by the authors.

Digital Object Identifier 10.1109/TSC.2023.3280914

I. Introduction

ATTRIBUTE-BASED anonymous credential provides a privacy-protecting identity authentication mechanism and is becoming an essential cryptographic primitive. An anonymous credential system consists of issuer, user, and verifier. A user obtains credentials from an issuer on a set of attributes that describe the user's access rights. The user then presents the credentials anonymously to a verifier by disclosing only a selected subset of the user's attributes or by proving that the user's attributes satisfy certain relations. Anonymous credentials protect user privacy by providing anonymity, and support fine-grained access control by providing selective disclosure and relationship proof.

Blockchain is a shared, decentralized ledger that helps record transactions in a business network. Deploying anonymous credentials in blockchain facilitates the auditing and tracing of assets. In blockchain platforms such as Ethereum [1] and Hyperledger [2], credentials can be issued through smart contracts, typically by integrating credential issuers into blockchain nodes for automated issuance. However, such systems generally assume that authenticity, confidentiality and availability are guaranteed even in the presence of a threshold number of dishonest or faulty nodes. Meeting this assumption, threshold credential schemes require only a certain number of issuers available online to implement credential issuing, making them suitable for deployment in blockchain platforms. Sonnino et al. [4] for the first time proposed such a threshold attributes-based anonymous credential scheme named Coconut and integrated it into Ethereum and Chainspace [3]. Coconut supports threshold issuance for both private and public attributes, multiple unlinkable presentations of anonymous credentials, selective disclosure of user attributes, and const-size credentials. However, Coconut does not support threshold tracing of user identities and threshold revocation of user credentials. The communication and computation complexities of credential verification in Coconut increase linearly with the number of each user's attributes, which is unfavorable for applications with massive user attributes.

Identity tracing and credential revocation is a necessary feature to enhance anonymous credentials schemes so that illegal and malicious user behavior can be investigated while user privacy is protected due to the use of anonymous credentials. It can help governments and enterprises achieve internal governance such as identity management, data auditing, and accountability. To prevent abuse of tracing and revocation rights, it is desirable that tracing and revocation be distributed to multiple entities and enabled as long as more than a threshold number of entities are cooperative.

The communication and computation complexities of credential verification matter in large-scale applications. The complexities of Coconut for verifying a user's credential increase linear with the number of the user's attributes, regardless of how many user's attributes are disclosed for credential verification. Such complexities are not ideal for applications in which massive user attributes are issued. For example, Anonymous Credentials (IRMA) [6] developed by the Privacy By Design Foundation promotes a wide range of real-world attributes to be used by governments and businesses, including diplomas, passports, cards, and membership IDs for online services, which are beyond personal attributes such as name, gender, address, and position, as well as attributes related to a user's computing devices such as hardware platform and software configuration. The number of attributes for certain users may grow to hundreds, making it necessary to reduce the complexities of verifying user's credentials in practice.

Addressing these issues, we propose a threshold attribute-based credential with redactable signature. Our scheme retains all the excellent features of the Coconut, such as supporting threshold issuing for both private and public attributes, multiple unlinkable presentations of anonymous credentials, selectively disclosing user attributes, and const-size credentials. In addition, our scheme supports threshold tracing of user identities and threshold revocation of user credentials to make it is more suitable for deployment into anonymous-yet-accountable distributed applications; we also reduce the complexities of selectively disclosing attributes with polynomial-based unlinkable redactable signatures (URS); finally, we formally prove our scheme is secure and implement it.

A. Contributions

Besides what Coconut has achieved, our scheme makes the following unique contributions:

• *Threshold tracing of user identities and threshold revocation of user credentials*: Our scheme supports threshold identity tracing and threshold credential revocation, such that the identity tracing and credential revocation capability are distributed to n_T tracers, and the user identity can be traced and credential can be revoked only when at least t_T-out-of-n_T tracers are cooperative. This avoids malicious tracing and malicious revocation caused by less than t_T tracers.

• *Efficient selective disclosure*: Our scheme improves the efficiency of credential verification significantly. Unlike Coconut [4], we do not rely on complex ZKP to prove the validity of user credentials in compliance with a selective disclosure policy. We innovatively replace costly ZKP with polynomial-based URS signature. We compare the communication and computational overheads of our scheme with Coconut and discover that the computation and communication overheads of credential verification are reduced significantly to $O(k)$ from $O(q)$, where q and k denote the total number of a user's attributes and the number of a user's disclosed attributes, respectively. In practice, the latter (k) is (significantly) smaller than the former (q).

• *Formal security proof*: We provide formal security proofs for our scheme, including anonymity, blindness, traceability, and non-frameability. We show how to reduce them to either well-known complexity assumptions or the security of proven cryptography primitives.

• *Application and implement*: Our scheme is suitable for deployment in blockchain and other anonymous-yet-accountable scenarios. In particular, we highlight its application in an accountable anonymous reporting system and an existing permissioned token system in *Supplementary Material A*, available online. We implement our scheme on a personal laptop and compare its performance with Coconut. In the credential verification algorithm, the communication cost of our scheme is 24.8% and 4.3%, and the computation cost of our scheme is 71.3% and 12.7% of Coconut for $q = 100$ and $q = 700$, respectively. In addition, we implemented the smart contract of our scheme on the blockchain platform and tested its cost to demonstrate its feasibility.

B. Related Work

1) Selective Disclosure Credentials: Anonymous attribute-based credentials supporting selective disclosure of attributes can be obtained in a similar way as using randomizable signatures: each user receives a signature as credential on (commitments to) a vector of attributes from a central authority; when the credential is shown, the user chooses a subset of their attributes to disclose, randomizes their signature (so that the generated signature and the issued signature cannot be linked) and proves the correspondence of this signature to the disclosed attributes as well as the undisclosed attributes in zero-knowledge proofs [8], [9], [10], [11], [12], [13]. In terms of privacy, this solution is entirely satisfactory. However, in terms of efficiency, it is not as satisfactory because undisclosed attributes are hidden in the proof of knowledge, which costs more than the user's disclosed attributes.

Fuchsbauer and Hanser [15], [16] randomized both the signature and the signed message (which is a commitment to the set of user's attributes) using a structure-preserving signature on equivalence classes (SPS-EQ) and then relying on subset-opening of set commitments (SC) for selective disclosure of attributes. They thereby avoided costly ZKP over hidden attributes. Unfortunately, once attributes are signed in their solution, one can disclose them but cannot prove that hidden attributes satisfy certain relationships. Lysyanskaya et al. [17], [18] were inspired by the paper of [16] on SPS-EQ. They provided a construction of delegatable anonymous credentials based on mercurial signature which can be used to randomize signatures, messages, and public keys. Their construction is not based on zero-knowledge proofs, and is therefore considerably more efficient. Unfortunately, the credentials constructed using this approach suffer from the same disadvantages as [16].

In [19], a new unlinkable redactable signature (URS) was constructed by Camenisch et al. and it allows users to redact part of a signed message, but still can prove that its signature is valid for a set of disclosed attributes. Unfortunately, their scheme can only be instantiated by Groth-Sahai proofs [20], and it is hard to compete with the most effective solution in practice. In [21], Sanders followed the approach based on URS from [19] and constructed a flexible redactable signature scheme that achieves unlinkability at almost no cost. Unlike the methods given in [16], [17], [18], Sanders's URS can prove complex relationships among attributes and does not rely on zero-knowledge proofs for partial verification. Unfortunately, the construction in [21] suffers from large public key size, which is, quadratic in q (i.e., a total number of a user's attributes). In [7], Sanders proposed an improved version of the construction of [21], which reduces the public key size to linear of q.

2) Distributed and Traceable Credentials: Due to its flexible deployment, privacy protection, and auditability, many distributed and traceable credential schemes have been proposed. Garman et al. [37] presented a decentralized anonymous credentials system for distributed ledgers; their system can be used to issue publicly verifiable claims without central issuers; however, showing credentials requires expensive double discrete logarithm proofs. Yang et al. [38] avoided complex double discrete logarithmic proofs using blacklist-based authentication. Unfortunately, none of the above schemes supports threshold issuing of credentials and threshold tracing of identities.

Hébant et al. [39] proposed a scheme of traceable constant-size multi-authority credentials based on aggregatable signatures with randomizable tags (ART-sign). Their scheme allows traceability and, for the first time, the cost for users to prove credentials is independent of the number of a user's attributes and the number of credential issuers. On the other hand, it shares the same disadvantage as in [16] which does not support proof of relationship of attributes.

Sonnino et al. [4] proposed "Coconut," which is a threshold anonymous credential system from Pointcheval–Sanders (PS) [13] signature, and Rial et al. [5] analyze the security properties of Coconut in the universal composability framework. Unfortunately, Coconut does not address the issues of threshold tracing of user's identities, and the efficiency of credential verification increases linearly with the number of each user's attributes.

There are also group signature schemes that implement threshold issuance and threshold tracing, but they are not attribute-based and cannot achieve flexible access control. Gennaro et al. [40] proposed to extend BBS [12] and CL [10] group-signature schemes to support threshold issuance. But their schemes follow the sign-and-encrypt paradigm, which results in a huge signature size to achieve threshold tracing. Camenisch et al. [14] recently proposed a short threshold dynamic group signature scheme. They formalized threshold dynamic group signatures, defined their security in the presence of multiple issuers and tracers, and presented an efficient, provably secure instantiation based on PS signatures [13].

II. Preliminary

A. Bilinear Pairing

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p. Let g and $\widetilde{g}$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. The mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map if it has three properties: (1) *bilinearity:* $\forall g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, we have $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$. (2) *non-degeneracy:* $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$. (3) *computability:* e can be efficiently computed. Our scheme is based on the Type-III pairing [27], which means that there is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

B. Computational Assumptions

$\mathsf{Discrete\ Logarithm\ (DL)\ assumption}$. Let $\mathbb{G}$ be an cyclic group of prime order and g is a generator of $\mathbb{G}$. Given $(g, g^x) \in \mathbb{G}^2$, the DL assumption holds in $\mathbb{G}$ if no efficient adversary can compute x with non-negligible probability.

$\mathsf{Decisional\ Diffie-Hellman\ (DDH)\ assumption}$ [28]: Let $\mathbb{G}$ be an cyclic group of prime order and g is a generator of $\mathbb{G}$. Given $(g, g^x, g^y, g^z) \in \mathbb{G}^4$, the DDH assumption holds in the group $\mathbb{G}$ if no efficient adversary can distinguish $z = x \cdot y$ from an element z that is randomly chosen from $\mathbb{Z}_p$.

C. Polynomial-Based Unlinkable Redactable Signatures

Polynomial-based unlinkable redactable signatures (URS) [7], which consists of a tuple (Setup,KeyGen, Sign, Derive, Verify) algorithms, is adopted to issue a credential for the attributes of a user.

- $\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, this algorithm generates public parameters $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \widetilde{g}, p, e)$.
- $\mathsf{KeyGen}(n)$: On input an integer n, this algorithm selects $(x, y) \leftarrow R\,\mathbb{Z}_p^*$ and computes $\widetilde{X} = \widetilde{g}^x$; $\widetilde{Y}_i = \widetilde{g}^{y^i} (i \in [1, n])$ and $Y_i = g^{y^i} (i \in [1, n] \cup [n + 2, 2n])$. The secret key is $sk = (x, y)$, and the public key is $pk = (\widetilde{X}, \{Y_i, \widetilde{Y}_i\}_{i=1}^n, \{Y_i\}_{i=n+2}^{2n})$.
- $\mathsf{Sign}(sk, \{m_i\}_{i=1}^n)$: To sign n messages $m_1, \ldots, m_n$ ($m_i \in \mathbb{Z}_p, i \in [1, n]$), this algorithm selects $\sigma_1 \leftarrow R\,\mathbb{G}_1$, computes $\sigma_2 = \sigma_1^{x+\sum_{i=1}^n y^i \cdot m_i}$. It sets $\sigma_3 = 1_{\mathbb{G}_1}$ and $\widetilde{\sigma} = 1_{\mathbb{G}_2}$, and outputs $\sigma = (\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$.
- $\mathsf{Derive}(pk, \sigma, \{m_i\}_{i=1}^n, \mathcal{D})$: On input a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$ on $\{m_i\}_{i=1}^n$, the public key pk and a index subset $\mathcal{D} \subseteq [1, n]$, this algorithm selects $(r, t) \leftarrow R\,\mathbb{Z}_p^*$ and computes $\sigma_1' = \sigma_1^r$; $\sigma_2' = \sigma_2^r (\sigma_1')^t$; $\widetilde{\sigma}' = \widetilde{g}^t \prod_{i \in [1,n] \setminus \mathcal{D}} \widetilde{Y}_i^{m_i}$. Then, for all $i \in \mathcal{D}$, it computes $c_i = H(\sigma_1', \sigma_2', \widetilde{\sigma}', \mathcal{D}, i)$ that is used to computes: $\sigma_3' = \prod_{i \in \mathcal{D}} [Y_{n+1-i}^t \cdot \prod_{j \in [1,n] \setminus \mathcal{D}} Y_{n+1-i+j}^{m_j}]^{c_i}$. It returns the derived signature $\sigma' = (\sigma_1', \sigma_2', \sigma_3', \widetilde{\sigma}')$ on $\{m_i\}_{i \in \mathcal{D}}$.
- $\mathsf{Verify}(pk, \sigma, \{m_i\}_{i \in \mathcal{D}})$: A signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \widetilde{\sigma})$ on $\{m_i\}_{i \in \mathcal{D}}$ is valid if the following equations hold: $e(\sigma_1, \widetilde{X}\widetilde{\sigma} \cdot \prod_{i \in \mathcal{D}} \widetilde{Y}_i^{m_i}) = e(\sigma_2, \widetilde{g})$ and $e(\prod_{i \in \mathcal{D}} Y_{n+1-i}^{c_i}, \widetilde{\sigma}) = e(\sigma_3, \widetilde{g})$, where $c_i = H(\sigma_1', \sigma_2', \widetilde{\sigma}', \mathcal{D}, i)$. If these equations are all satisfied, then the algorithm returns 1; otherwise returns 0.

In the random oracle and generic group models, URS [7] signature is unforgeable and unlinkable.

D. ElGamal Encryption

ElGamal encryption [22] scheme, which consists of a tuple (Setup, KeyGen, Enc, Dec) algorithms, is adopted to to blind private attributes and keys.

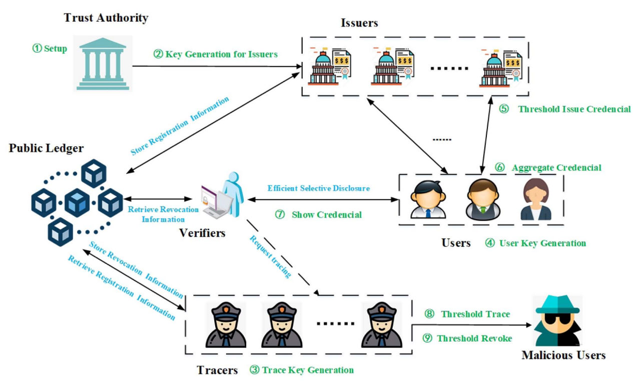


Fig. 1. Architecture.

• $\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, this algorithm generates a cyclic group $\mathbb{G}$ with prime order p. It chooses a generator g of $\mathbb{G}$, and outputs public parameters $pp = (\mathbb{G}, g, p)$.

• $\mathsf{KeyGen}(pp)$. This algorithm chooses a random $z \leftarrow R\,\mathbb{Z}_p^*$ as a secret key, and outputs the public key $Z = g^z$.

• $\mathsf{Enc}(Z, m)$: On input a public key Z and a message m, this algorithm chooses a random $r \leftarrow R\,\mathbb{Z}_p^*$, and encrypts m as $C = (C_1, C_2) = (g^r, Z^r m)$.

• $\mathsf{Dec}(z, C)$: This algorithm decrypts the message m as $m = C_2 C_1^{-z}$.

ElGamal encryption [22] scheme is IND-CPA secure under DDH assumption. In our construction, it is easy to generate ZKP with the ElGamal encryption.

E. Signature of Knowledge

Signature of knowledge (SoK) [30] for a NP-relation $\mathcal{R}$ with the language $L_{\mathcal{R}} = \{y : \exists x, (x, y) \in \mathcal{R}\}$, which consists of a tuple (Setup, Sign, Verify) algorithms, is adopted to prove the knowledge of private attributes and keys.

• $\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, this algorithm outputs a public parameter pp.

• $\mathsf{Sign}(m, x, y)$: On input a message m and a relation $(x, y) \in \mathcal{R}$, it outputs a SoK: $\Pi = \mathsf{SoK}\{x : (x, y) \in \mathcal{R}\}$.

• $\mathsf{Verify}(m, \Pi, y)$: On input a message m, a SoK Π and a statement y. If π is valid, return 1; otherwise return 0.

A SoK is $SimExt$ secure [30] if it satisfies correctness, simulatability and extractability. The SoK can be instantiated by Fiat-Shamir paradigm [29] incorporated with zero-knowledge protocols in [31].

III. System and Security Model

A. System Model

As shown in Fig. 1, the architecture of our scheme consists of five types of parties: a one-time trusted third party (TTP), multiple issuers (I), multiple tracers (T), users (U) and verifiers (V). In addition, it contains a public append-only ledger ($\mathcal{L}$) for storing and retrieving user registration/revocation information. The specific role of each party is described as follows.

• TTP is a one-time trusted third party responsible for setting up the system (see step ①) and generating private/public key pairs for n_I issuers (②).

• I_i is an independent issuer responsible for issuing partial credentials for users (⑤). There are n_I issuers in a distributed system. It is reasonable to assume that some of the issuers are malicious or can be compromised and that some of the issuers may not be available online all the time. It is thus required that the issuers can issue credentials to users (⑤) as long as at least t_I honest issuers are online.

• T_i is an independent tracer responsible for tracing user identities (⑧) and revoking user credentials (⑨). There are n_T tracers in a distributed system. It is reasonable to assume that some of the tracers are malicious or can be compromised and that some of the tracers may not be available online all the time. It is thus required that the tracers can trace users' identities (⑧) as long as at least t_T honest tracers are online. If a user is discovered with malicious behavior, t_T honest tracers jointly revoke the issued credentials (⑨).

• U with a set of private attributes should register to t_I-out-of-n_I issuers (⑤), and then the acquired t_I partial credentials can be aggregated into single (⑥). When showing credential (⑦), U needs to disclose a subset of attributes or provide proof of relationship of the hided attributes to V.

• V is an honest but curious credential verifier that honestly validates the token generated by the user during credential verification (⑦) but is curious to know users' undisclosed attributes and real identities. V requests the tracers to trace the identities of malicious users who do not comply with the access policy.

• $\mathcal{L}$ is a public append-only ledger. During user registration (⑤), issuers store user registration information to $\mathcal{L}$. During user tracing (⑧), tracers retrieve user registration information from $\mathcal{L}$. During user revocation (⑨), tracers store user revocation information to $\mathcal{L}$. During credential verification (⑦), verifiers retrieve user revocation information from $\mathcal{L}$. $\mathcal{L}$ can be instantiated using blockchain, where issuers/tracers can store user registration/revocation information through the blockchain's transaction protocol.

B. Formal Definition

The notations used in our scheme are summarized in Table I, and the algorithms are formally defined below. Since $\mathcal{L}$ is publicly available to all participants, the following algorithm definitions take $\mathcal{L}$ as input by default.

• $\mathsf{Setup}(1^\lambda, n_I, t_I, n_T, t_T, q) \rightarrow pp$: This algorithm is operated only once by TTP. It inputs a security parameter λ, a number n_I of issuers and threshold value t_I, a number n_T of tracers and threshold value t_T, and a total number q of user attributes, and outputs the system parameter pp.

• $\mathsf{TTPKeyGen}(pp) \rightarrow (pk, \{isk_i, ipk_i\}_{i=1}^{n_I})$: This algorithm is operated by TTP. It inputs the system parameter pp, and outputs the verification key pk. It then generates private key isk_i and public key ipk_i for each I_i, where $i \in [1, n_I]$.

• $\mathsf{TraceKeyGen}(pp) \rightarrow (tsk_i, tpk_i)$: This algorithm is operated by each T_i ($i \in [1, n_T]$) that inputs the system parameter pp, and outputs private key tsk_i and public key tpk_i.

TABLE I
SUMMARY OF NOTATIONS

Notation	Description
$1^\lambda/\epsilon(\lambda)$	security parameter/negligible function
n_I/t_I	a number of issuers/thresholds value
n_T/t_T	a number of tracers/thresholds value
q/k	a number of a user/disclosed attributes
$[1,n]$	the set $\{1,2,\cdots,n\}$
$\mathcal{D}$	a subset of $[1,q]$
$\mathcal{I}$	a subset of $[1,n_I]$, $\lvert\mathcal{I}\rvert=t_I$
$\mathcal{T}$	a subset of $[1,n_T]$, $\lvert\mathcal{T}\rvert=t_T$
$x \xleftarrow{R} \mathbb{Z}_p$	x is randomly selected from the set $\mathbb{Z}_p$
$\mathsf{TTP}/\mathcal{L}$	trusted third party/public append-only ledger
U/V	user/verifier
$\mathsf{I}_i/\mathsf{T}_i$	the i issuer/tracer
id	a user's identity information
reg/rev	a user's registration/revocation information
pp/pk	public parameters/verification public key
$\{m_j\}_{j=1}^q$	a set of a user's attributes
usk/upk	secret/public key of U
isk_i/ipk_i	secret/public key of I_i
tsk_i/tpk_i	secret/public key of T_i
$cred_i/cred$	a partial/aggregated credential of U
tok	a token of credential
$\mathcal{H}_1:\{0,1\}^*\to\mathbb{Z}_p$	a collision resistant hash function
$\mathcal{H}_2:\{0,1\}^*\to\mathbb{G}_1$	a collision resistant hash function
$\perp$	failure identifier

• $\mathsf{UKeyGen}(id,pp)\to(usk,upk)$: This algorithm is operated by U that inputs the user identity id and the system parameter pp, and outputs private key usk and public key upk.

• $\langle\mathsf{Obtain}(id,usk,upk,\{m_j\}_{j=1}^q,ipk_i,\{tpk_j\}_{j=1}^{n_T},pp)\leftrightarrow\mathsf{Issue}\ (isk_i,ipk_i,pp)\rangle\to(cred_i/\perp)$: This algorithm is operated by interacting between U and I_i. To obtain a partial credential from I_i, U takes user identity id, private/public key pair (usk,upk), private attributes $\{m_j\}_{j=1}^q$, I_i's public key ipk_i, tracers' public key $\{tpk_j\}_{j=1}^{n_T}$ and pp as inputs. I_i takes private key isk_i, public key ipk_i, and pp as inputs. If the execution fails, the algorithm returns $\perp$, otherwise the user registration information reg is stored to $\mathcal{L}$ and the user partial credential $cred_i$ is returned.

• $\mathsf{CredAgg}(\{cred_i\}_{i\in\mathcal{I}},pp)\to cred$: This algorithm is operated by U that inputs pp and t_I partial credentials $\{cred_i\}_{i\in\mathcal{I}}$, where $\mathcal{I}\subset[1,n_I]$, and outputs an aggregated credential $cred$.

• $\langle\mathsf{Show}(usk,\{m_j\}_{j=1}^q,\mathcal{D},cred,pp)\leftrightarrow\mathsf{Verify}(pk,pp)\rangle\to(0/1,tok)$: This algorithm is operated by interacting between U and V. U takes usk, $\{m_j\}_{j=1}^q$, a index subset $\mathcal{D}$ ($\mathcal{D}\in[1,q]$) of disclosed attributes, $cred$ and pp as inputs. V takes pk and pp as inputs. At the end of this algorithm, V outputs a bit b (outputs 1 if $cred$ is valid and not revoked or 0 otherwise) and a credential token tok.

• $\langle\mathsf{Trace}(tsk_i,tok,pp)\rangle_{i\in\mathcal{T}}\to(id/\perp)$: This algorithm is jointly operated by t_T tracers $\{\mathsf{T}_i\}_{i\in\mathcal{T}}$ ($\mathcal{T}\subset[1,n_T]$) who input private keys tsk_i, tok and pp. It traverses the registration information in $\mathcal{L}$, returns the id of the user who has computed a valid credential token tok, and returns $\perp$ if the execution fails.

• $\langle\mathsf{Revoke}(tsk_i,id,pp)\rangle_{i\in\mathcal{T}}\to rev$: This algorithm is jointly operated by t_T tracers $\{\mathsf{T}_i\}_{i\in\mathcal{T}}$ ($\mathcal{T}\subset[1,n_T]$) who input private keys tsk_i, user identity id and pp. It outputs the user id's revocation information rev, and stores the revocation information to $\mathcal{L}$.

Correctness: Our scheme is correct if (1) a tok of a credential $cred$ with respect to a selective disclosure set $\{m_j\}_{j\in\mathcal{D}}$ should be accepted by the credential verification algorithm, provided that $cred$ on a set of attributes $\{m_j\}_{j=1}^q$ are aggregated from partial credentials issued by t_I honest issuers and not revoked; (2) by operating the tracing and revocation algorithm, any t_T tracers should all output the same user identity id and revocation information rev. The formal correctness definition is shown in *Supplemental Material B*, available online.

C. Security Model

Our scheme should satisfy the following security requirements: anonymity, blindness, traceability and non-frameability. The security model is defined following the works in [7], [14], [15], [21], and we provide formal definitions of security requirements. All security definitions use the following global variables and oracles.

Global Variables: In the security games, global variables are maintained by the challenger and can be accessed by all oracles. $\mathcal{HU}$: the set of honest users identities; $\mathcal{CU}$: the set of corrupt users identities; $\mathcal{CRED}$: the set that stores $(id,\{m_j\}_{j=1}^q,cred,usk)$ each time a credential is issued for user id; $\mathcal{TK}$: the set that stores (id,tok) each time a user id computes a token tok of credential; $\mathcal{TR}$: the set that stores (id,tok) each time a tok is opened by tracers.

Oracles: Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary.

$\mathcal{O}_{\mathsf{ObtIss}}(id,\mathcal{I},\{m_j\}_{j=1}^q)$. It is an oracle that can be used to issue a credencial for an honest users id with the attribute set $\{m_j\}_{j=1}^q$. If $id\in\mathcal{CU}$ or $\mathcal{I}\subsetneq[1,n_I]$, it returns $\perp$. Otherwise, it generates key pair for id by running: $(usk,upk)\leftarrow\mathsf{UKeyGen}(id,pp)$, and t_I issuers issue partial credentials to id by running: $\{\langle\mathsf{Obtain}(id,\ usk,\ upk\ \{m_j\}_{j=1}^q,\ ipk_i,\{tpk_j\}_{j=1}^{n_T},\ pp)\leftrightarrow\mathsf{Issue}(isk_i,ipk_i,\ pp)\rangle\to cred_i\}_{i\in\mathcal{I}}$. After obtaining t_I partial credentials, it aggregates credentials by running: $\mathsf{CredAgg}(\{cred_i\}_{i\in\mathcal{I}},pp)\to cred$: It adds id to $\mathcal{HU}$ and appends $(id,\{m_j\}_{j=1}^q,cred,usk)$ to $\mathcal{CRED}$.

$\mathcal{O}_{\mathsf{Obtain}}(id,\mathcal{I},\{m_j\}_{j=1}^q)$. It is an oracle that can be used to play the corrupt issuers jointly issue a credential for an honest user id with the attributes set $\{m_j\}_{j=1}^q$. If $id\in\mathcal{CU}$, it returns $\perp$. Otherwise, it generates key pair for id by running: $(usk,upk)\leftarrow\mathsf{UKeyGen}(id,pp)$ and obtains t_I partial credentials by running: $\{\langle\mathsf{Obtain}(id,usk,upk,\{m_j\}_{j=1}^q,ipk_i,\{tpk_j\}_{j=1}^{n_T},pp)\leftrightarrow\mathcal{A}(\cdot)\rangle\to cred_i\}_{i\in\mathcal{I}}$, where the issuer's side is executed by the adversary. After obtaining t_I partial credentials, it aggregates credentials by running: $\mathsf{CredAgg}(\{cred_i\}_{i\in\mathcal{I}},pp)\to cred$. It adds id to $\mathcal{HU}$ and appends $(id,\{m_j\}_{j=1}^q,cred,usk)$ to $\mathcal{CRED}$.

$\mathcal{O}_{\mathsf{Issue}}(id,\mathcal{I},\{m_j\}_{j=1}^q)$. It is an oracle that can be used to play the honest issuers jointly issue a credential for a corrupt user id with the attributes set $\{m_j\}_{j=1}^q$. If $id\in\mathcal{HU}$, it returns $\perp$. Otherwise, it obtains t_I partial credentials by running: $\{\langle\mathcal{A}(\cdot)\leftrightarrow\mathsf{Issue}(isk_i,ipk_i,pp)\rangle\to cred_i\}_{i\in\mathcal{I}}$, where the user's side is executed by the adversary. After obtaining t_I partial credentials, it aggregates credentials by running:

```
$Exp^{anon-b}(\mathcal{A},\lambda)$
1. $\mathcal{L}, \mathcal{HU}, \mathcal{CU}, \mathcal{CRED}, \mathcal{TK}, \mathcal{TR} \leftarrow \emptyset$; $pp \leftarrow \mathsf{Setup}(1^{\lambda}, n_I, t_I, n_T, t_T, q)$;
2. $(st_1, \{i, isk_i\}_{i>t_I}, \{ipk_i\}_{i\in[1,n_I]}, pk) \leftarrow \langle \mathcal{A}(\mathsf{keygen}, pp), \{\mathsf{TTPKeyGen}(pp)\}_{i>t_I}\rangle$;
3. $(st_2, \{i, tsk_i\}_{i\geq t_T}, \{tpk_i\}_{i\in[1,n_T]}) \leftarrow \langle \mathcal{A}(st_1), \{\mathsf{TraceKeyGen}(pp)\}_{i\geq t_T}\rangle$;
4. $\mathcal{O} \leftarrow \{\mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{Show}}, (\mathcal{O}_{\mathsf{Trace}})_{\mathcal{T}\subset[t_T,n_T]}, \mathcal{O}_{\mathsf{AnCh}_b}\}$;
5. $b^* \leftarrow \mathcal{A}^{\mathcal{O}}(st_2)$;
6. If $\mathcal{TR}$ stores the output of $\mathcal{O}_{\mathsf{AnCh}_b}$, then return 0;
7. If $\mathcal{CU}$ stores an input of $\mathcal{O}_{\mathsf{AnCh}_b}$, then return 0;
8. If $b = b^*$, return 1, otherwise, return 0.
```

Fig. 2. Anonymity Security Game.

$\mathsf{CredAgg}(\{cred_i\}_{i\in\mathcal{I}}, pp) \rightarrow cred$. It adds id to $\mathcal{CU}$ and appends $(id, \{m_j\}_{j=1}^{q}, cred, *)$ to $\mathcal{CRED}$, where $*$ indicates the unknown private key.

$\mathcal{O}_{\mathsf{CU}}(id)$. It is an oracle that can be used to corrupt an honest user id. If $id \in \mathcal{CU}$, then it returns $\perp$. If $i \in \mathcal{HU}$, then it removes id from $\mathcal{HU}$ and adds id to $\mathcal{CU}$, and returns $(id, \{m_j\}_{j=1}^{q}, cred, usk)$.

$\mathcal{O}_{\mathsf{Show}}(id, \mathcal{D})$. It is an oracle that can be used to play a malicious verifier verify a token for an honest user id. If $id \notin \mathcal{HU}$, it returns $\perp$. Otherwise, it runs $\langle \mathsf{Show}(usk, \{m_j\}_{j=1}^{q}, \mathcal{D}, cred, pp) \leftrightarrow \mathcal{A}(\cdot)\rangle \rightarrow (1, tok)$, where the verifier's side is executed by adversary. It adds (id, tok) to $\mathcal{TK}$.

$\mathcal{O}_{\mathsf{Trace}}(tok, \mathcal{T})$. It is an oracle that can be used to play the honest tracers jointly trace an honest user. It runs the tracing algorithm: $\langle \mathsf{Trace}(tsk_i, tok)\rangle_{i\in\mathcal{T}} \rightarrow id$, and appends $(id, token)$ to $\mathcal{TR}$.

$\mathcal{O}_{\mathsf{AnCh}_b}(id_0, id_1, \mathcal{D})$. It is an oracle that takes as inputs the identity of two honest users who have the same user's attributes. It runs: $\langle \mathsf{Show}(usk_b, \{m_j\}_{j=1}^{q}, \mathcal{D}, cred_b, pp) \leftrightarrow \mathcal{A}(\cdot)\rangle \rightarrow (1, tok_b)$, and returns tok_b, where $b \in \{0,1\}$.

$\mathcal{O}_{\mathsf{BlCh}_b}(id, \{m_j^0\}_{j=1}^{q}, \{m_j^1\}_{j=1}^{q}, \mathcal{I})$. It is an oracle that takes as inputs two different sets of attributes $\{m_j^0\}_{j=1}^{q}, \{m_j^1\}_{j=1}^{q}$ for the honest user id. It runs: $\{\langle \mathsf{Obtain}(id, usk, upk, \{m_j^b\}_{j=1}^{q}, ipk_i, \{tpk_j\}_{j=1}^{n_T}, pp) \leftrightarrow \mathsf{Issue}(isk_i, ipk_i, pp)\rangle \rightarrow cred_{i,b}\}_{i\in\mathcal{I}}$. It aggregates credentials by running: $\mathsf{CredAgg}(\{cred_{i,b}\}_{i\in\mathcal{I}}, pp) \rightarrow cred_b$, and returns $cred_b$, where $b \in \{0,1\}$.

Anonymity: Anonymity can protects the user's privacy as long as fewer than t_T tracers collude; in such case, no verifier can learn any information about the user's attributes and secret key during the execution of the credential verification algorithm, except for the disclosed attributes. We even allow a token to be opened without affecting the anonymity of other tokens, thus ensuring both backward and forward unlinkability. Without loss of generality, we assume that the first $1, \ldots t_T - 1$ tracers and the first $1, \ldots, t_I$ issuers are controlled by adversary in the anonymity game.

Definition 1: The *anonymity* is defined by game Exp^{anon-b} in Fig. 2. Our scheme is anonymous, if for any PPT adversary $\mathcal{A}$, there is a negligible function $\epsilon(\lambda)$ such that

$$Adv^{anon} = |\Pr\left[Exp^{anon-1}(\mathcal{A},\lambda) = 1\right] - \frac{1}{2}| \leqslant \epsilon(\lambda)$$

```
$Exp^{blind-b}(\mathcal{A},\lambda)$
1.$\mathcal{L}, \mathcal{HU}, \mathcal{CU}, \mathcal{CRED}, \mathcal{TK}, \mathcal{TR} \leftarrow \emptyset$; $pp \leftarrow \mathsf{Setup}(1^{\lambda}, n_I, t_I, n_T, t_T, q)$;
2. $(st_1, pk, \{i, ipk_i\}_{i\in[1,n_I]}, pk) \leftarrow \mathcal{A}(\mathsf{keygen}, pp)$;
3. $\mathcal{O} \leftarrow \{\mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{BlCh}_b}\}$;
4. $b^* \leftarrow \mathcal{A}^{\mathcal{O}}(st_1)$;
5. If $b = b^*$, return 1, otherwise, return 0.
```

Fig. 3. Blindness Security Game.

```
$Exp^{trace}(\mathcal{A},\lambda)$
1.$\mathcal{L}, \mathcal{HU}, \mathcal{CU}, \mathcal{CRED}, \mathcal{TK}, \mathcal{TR} \leftarrow \emptyset$; $pp \leftarrow \mathsf{Setup}(1^{\lambda}, n_I, t_I, n_T, t_T, q)$;
2. $(st_1, \{i, isk_i\}_{i\geq t_I}, \{ipk_i\}_{i\in[1,n_I]}, pk) \leftarrow \langle \mathcal{A}(\mathsf{keygen}, pp), \{\mathsf{TTPKeyGen}(pp)\}_{i\geq t_I}\rangle$;
3. $(st_2, \{i, tsk_i\}_{i>t_T}, \{tpk_i\}_{i\in[1,n_T]}) \leftarrow \langle \mathcal{A}(st_1), \{\mathsf{TraceKeyGen}(pp)\}_{i>t_T}\rangle$;
4. $\mathcal{O} \leftarrow \{(\mathcal{O}_{\mathsf{ObtIss}})_{\mathcal{I}\subset[t_I,n_I]}, (\mathcal{O}_{\mathsf{Issue}})_{\mathcal{I}\subset[t_I,n_I]}, \mathcal{O}_{\mathsf{CU}}, (\mathcal{O}_{\mathsf{Show}}, \mathcal{O}_{\mathsf{Trace}})_{\mathcal{T}\subset[t_T+1,n_T]}\}$;
5. $tok \leftarrow \mathcal{A}^{\mathcal{O}}(st_2)$;
6. If $0 \leftarrow \mathsf{Verify}(tok, pk, pp)$ or $tok \in \mathcal{TR}$, then return 0;
7. If $id^* \leftarrow \langle \mathsf{Trace}(\mathcal{L}, i, tsk_i, tok)\rangle_{i\in\mathcal{T}}$ and $id^* \in \mathcal{CU}$, then return 0;
8. If $\perp \leftarrow \langle \mathsf{Trace}(\mathcal{L}, i, tsk_i, tok)\rangle_{i\in\mathcal{T}}$, then return 1;
9. Return 0.
```

Fig. 4. Traceability Security Game.

Blindness: Blindness ensures that issuers cannot learn any information about the user's private attributes during the execution of the credential issuing algorithm, except for the fact that the user has knowledge of these attributes. We define blindness in the malicious issuer model [15], [41] so that the adversary controls all issuers in the blindness security game.

Definition 2: The *blindness* is defined by game $Exp^{blind-b}$ in Fig. 3. Our scheme is blind, if for any PPT adversary $\mathcal{A}$, there is a negligible function $\epsilon(\lambda)$ such that

$$Adv^{blind} = |\Pr\left[Exp^{blind-1}(\mathcal{A},\lambda) = 1\right] - \frac{1}{2}| \leqslant \epsilon(\lambda)$$

Traceability: Traceability ensures that the tracing algorithm cannot return $\perp$ for any valid token. Following Camenisch et al.'s [14] definition, in traceable security games, the number of corrupt tracers can exceed t_T, allowing the adversary to trace arbitrary users, and the number of corrupt issuers is less than t_I, preventing the adversary from issuing credentials to honest users.

Definition 3: The *traceability* is defined by game Exp^{trace} in Fig. 4. Our scheme is traceable, if for any PPT adversary $\mathcal{A}$,

```
$Exp^{non-frame}(\mathcal{A}, \lambda)$
1.$\mathcal{L}, \mathcal{HU}, \mathcal{CU}, \mathcal{CRED}, \mathcal{TK}, \mathcal{TR} \leftarrow \emptyset$; $pp \leftarrow \mathsf{Setup}(1^{\lambda}, n_I, t_I, n_T, t_T, q)$;
2. $(st_1, \{i, isk_i\}_{i>t_I}, \{ipk_i\}_{i\in[1,n_I]}, pk) \leftarrow \langle \mathcal{A}(\mathsf{keygen}, pp), \{\mathsf{TTPKeyGen}(pp)\}_{i>t_I}\rangle$;
3. $(st_2, \{i, tsk_i\}_{i>t_T}, \{tpk_i\}_{i\in[1,n_T]}) \leftarrow \langle \mathcal{A}(st_1), \{\mathsf{TraceKeyGen}(pp)\}_{i>t_T}\rangle$;
4. $\mathcal{O} \leftarrow \{\mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{Show}}, (\mathcal{O}_{\mathsf{Trace}})_{\mathcal{T}\subset[t_T+1,n_T]}\}$
5. $tok \leftarrow \mathcal{A}^{\mathcal{O}}(st_2)$;
6. If $0 \leftarrow \mathsf{Verify}(tok, pk, pp)$, then return 0;
7. If $tok \in \mathcal{TK}$ or $tok \in \mathcal{TR}$, then return 0;
8. If $id^* \leftarrow \langle \mathsf{Trace}(\mathcal{L}, i, tsk_i, tok)\rangle_{i\in\mathcal{T}}$ and $id^* \in \mathcal{HU}$,
   then return 1.
9. Return 0.
```

Fig. 5. Non-frameability Security Game.

there is a negligible function $\epsilon(\lambda)$ such that

$$Adv^{trace} = |\Pr\left[Exp^{trace}(\mathcal{A}, \lambda)\right] = 1| \leqslant \epsilon(\lambda)$$

Non-frameability: Non-frameability ensures that even if more than t_I issuers and t_T tracers collude, they cannot falsely claim that any honest user generated any valid token but in fact the users did not generate the token by themselves. This notion captures the unforgeability property required for threshold credentials and guarantees that only honest users can compute valid tokens.

Definition 4: The *non-frameability* is defined by game $Exp^{non-frame}$ in Fig. 5. Our scheme is framing-resistance, if for any PPT adversary $\mathcal{A}$, there is a negligible function $\epsilon(\lambda)$ such that

$$Adv^{non-frame} = |\Pr\left[Exp^{non-frame}(\mathcal{A}, \lambda)\right] = 1| \leqslant \epsilon(\lambda)$$

IV. Our Construction

A. High-Level Overview

As shown in Fig. 1, the working flow of our scheme is described as follows. TTP sets up the system parameters (①), and generates private/public keys for n_I issuers (②). Each tracer T_i $(i \in [1, n_T])$ generates a private/public key pair to trace user identities (③). When joining the system, U should generate a private/public key pair (④), and then U initiates a request to n_I issuers to obtain an attribute-based credential. Each issuer (online) issues a partial credential (⑤) and stores the user's registration information in $\mathcal{L}$. After receiving credentials from t_I issuers, U aggregates them to form into a single credential (⑥). When U needs to show the credential, U can re-randomize it and efficiently disclose a subset of attributes (⑦). If illegal tokens generated by U need to be traced, any t_T tracers (online) jointly trace the user's identity (⑧). If a user is discovered with malicious behavior, t_T tracers (online) jointly revoke the user's credentials (⑨) and store the revocation information in $\mathcal{L}$.

Our scheme is inspired by the concepts in Sanders's URS signature [7], Camenisch et al. 's threshold dynamic group signature [14], Shamir's secret sharing scheme [23] and ElGamal encryption [22]. The challenge is to combine and adapt them such that our scheme supports threshold tracing of users' identities, threshold revocation of users' credentials, threshold issuing of private attributes, and efficient selective disclosure.

First, we adopted the threshold idea of Camenisch et al. and adapted it to make the tracing algorithm and revocation algorithm of our scheme more efficient. (1) In credential issuing algorithm, a user divides their private key into n_T shares using Shamir's secret sharing scheme and encrypts each share using ElGamal's encryption scheme, then the ciphertexts of all shares are sent to all issuers as part of the registration information. (2) In the credential showing algorithm, the user has to compute the signature of knowledge of their private key. (3) In tracing algorithm, the tracers use a pairing-based equation to determine whether the private key hidden in the signature of knowledge matches the private key hidden in the registration information. (4) In revocation algorithm, the tracers decrypt the share of the user's private key and jointly recover the complete revocation information, from which the verifier can determine whether the user's credentials have been revoked according to a pairing-based equation.

Second, our scheme incorporates Sander's polynomial-based URS signature in credential issuance. Sander's URS signature is chosen because it is highly efficient for selective disclosure of attributes, it supports relationship proofs of hidden attributes, and its public key is linear in q. To support threshold issuance, our scheme relies on Shamir's secret sharing scheme, which implies that all issuers' key generation is performed by TTP. This limitation can also be mitigated by performing key generation in a distributed manner [24], [25] instead of TTP, but this is inefficient.

Finally, ElGamal's encryption scheme is also used to implement the issuance of private attributes because of its homomorphic and supporting efficient zero-knowledge proof. In the credential issuing algorithm, a user uses the ElGamal algorithm to encrypt each private attribute and computes the signature of knowledge of each ciphertext to prove the correctness of all ciphertexts. After receiving the signature of ciphertexts, the user uses its homomorphism to unblind the signature and obtain the credential of private attributes.

B. Concrete Construction

$\mathsf{Setup}(1^{\lambda}, n_I, t_I, n_T, t_T, q) \rightarrow pp$: TTP takes a security parameter λ, a number n_I of issuers and threshold value t_I, a number n_T of tracers and threshold value t_T, and a number q of user's attributes as inputs to create the system parameters pp. TTP generates Type-III bilinear pair parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \widetilde{g}, p, e)$ and selects $(h_1, h_2, \ldots, h_{q+1}) \leftarrow R\, \mathbb{G}_1$. The system parameters are $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \widetilde{g}, p, e, n_I, t_I, n_T, t_T, q, h_1, h_2, \ldots, h_{q+1})$.

$\mathsf{TTPKeyGen}(pp) \rightarrow (pk, \{isk_i, ipk_i\}_{i=1}^{n_I})$: As shown in Fig. 6, TTP takes pp as input to generate verification key pk

$\mathsf{TTPKeyGen}(pp) \rightarrow (pk, \{isk_i, ipk_i\}_{i=1}^{n_I})$

Select two random $(x, y) \xleftarrow{R} \mathbb{Z}_p$, and compute $\widetilde{X} = \widetilde{g}^x$; $\widetilde{Y}_j = \widetilde{g}^{y^j}, j \in [1, q+1]$; $Y_j = g^{y^j}, j \in [1, q+1] \cup [q+3, 2(q+1)]$.
Choose $2(q+1)$ polynomials of degree $(t_I - 1)$ with cofficients in $\mathbb{Z}_p^*$, denoted $f_x, f_{y_1}, f_{y_2}, \cdots, f_{y_{q+1}}, f_{y_{q+3}}, \cdots, f_{y_{2(q+1)}}$, and set $f_x(0) = x, f_{y_1}(0) = y, f_{y_2}(0) = y^2, \cdots, f_{y_{q+1}}(0) = y^{q+1}, f_{y_{q+3}}(0) = y^{q+3}, \cdots, f_{y_{2(q+1)}}(0) = y^{2(q+1)}$.
For each $i \in [1, n_I]$, compute $x_i = f_x(i)$; $\{y_{i,j} = f_{y_j}(i)\}_{j=1}^{q+1}$; $\{y_{i,j} = f_{y_j}(i)\}_{j=q+3}^{2(q+1)}$ and $\widetilde{X}_i = \widetilde{g}^{x_i}$; $\{Y_{i,j} = g^{y_{i,j}}\}_{j=1}^{q+1}$; $\{Y_{i,j} = g^{y_{i,j}}\}_{j=q+3}^{2(q+1)}$; $\{\widetilde{Y}_{i,j} = \widetilde{g}^{y_{i,j}}\}_{j=1}^{q+1}$.
The verification key $pk = (\widetilde{X}, \{Y_j\}_{j=1}^{q+1}, \{Y_j\}_{j=q+3}^{2(q+1)}, \{\widetilde{Y}_j\}_{j=1}^{q+1})$. For each $i \in [1, n_I]$, secret key $isk_i = (x_i, y_{i,1}, y_{i,2}, \ldots, y_{i,q+1}, y_{i,q+3}, \ldots, y_{i,2(q+1)})$, public key $ipk_i = (\widetilde{X}_i, \{Y_{i,j}\}_{j=1}^{q+1}, \{Y_{i,j}\}_{j=q+3}^{2(q+1)}, \{\widetilde{Y}_{i,j}\}_{j=1}^{q+1})$.

Fig. 6. TTP Key Generation Algorithm.

$\left\langle \mathsf{Obtain}(id, usk, upk, \{m_j\}_{j=1}^{q}, ipk_i, \{tpk_j\}_{j=1}^{n_T}, pp) \leftrightarrow \mathsf{Issue}(isk_i, ipk_i, pp) \right\rangle \rightarrow cred_i$

User U | Issuer I_i

Select $z \xleftarrow{R} \mathbb{Z}_p$, and compute ElGamal public key $Z = g^z$.
Compute $h = \mathcal{H}_2(id)$, for each $j \in [1, q]$, selects $r_j \xleftarrow{R} \mathbb{Z}_p$, compute $\alpha_j = g^{r_j}$, $\beta_j = Z^{r_j} h^{m_j}$.
Select $(d_1, d_2, \ldots, d_{t_T-1}) \xleftarrow{R} \mathbb{Z}_p$, and set polynomial of $t_T - 1$ degrees: $f_{usk}(x) = \sum_{j=1}^{t_T-1} d_j x^j + usk$.
For each $i \in [1, n_T]$, selects $k_i \xleftarrow{R} \mathbb{Z}_p$, compute $s_i = f_{usk}(i), \widetilde{C}_{1,i} = \widetilde{g}^{k_i}, \widetilde{C}_{2,i} = (tpk_i)^{k_i} \widetilde{Y}_{q+1}^{s_i}$.
For each $j \in [1, t_T - 1]$, compute $D_j = h^{d_j}$.
Compute the proof Π_1:
$\mathsf{SoK}\{(z, m_1, \ldots, m_q, usk, r_1, \ldots r_q, k_1, \ldots, k_{n_T}) : Z = g^z, \{\alpha_j = g^{r_j}, \beta_j = Z^{r_j} h^{m_j}\}_{j=1}^{q}, upk = h^{usk}, \{\widetilde{C}_{1,i} = \widetilde{g}^{k_i}, e(h, \widetilde{C}_{2,i}(tpk_i)^{-k_i}) = e(upk \cdot \prod_{j=1}^{t_T-1} D_j^{i^j}, \widetilde{Y}_{q+1})\}_{i=1}^{n_T}\}$.
Set $reg = (id, Z, h_{usk}, \{\alpha_j, \beta_j\}_{j=1}^{q}, \{\widetilde{C}_{1,i}, \widetilde{C}_{2,i}\}_{i=1}^{n_T}, \{D_j\}_{j=1}^{t_T-1}, \Pi_1)$.

$\xrightarrow{reg}$

Recompute $h = \mathcal{H}_2(id)$.
Verify the proof Π_1.
Compute $\alpha_i' = \prod_{j=1}^{q} \alpha_j^{y_{i,j}}$, $\beta_i' = h^{x_i} \prod_{j=1}^{q} \beta_j^{y_{i,j}} upk^{y_{i,q+1}}$.
Store reg into $\mathcal{L} \leftarrow reg$.

$\xleftarrow{\alpha_i', \beta_i'}$

Compute $\sigma_i = \beta_i'(\alpha_i')^{-z}$, verify $e(\sigma_i, \widetilde{g}) = e(h, \widetilde{X}_i \prod_{j=1}^{q} \widetilde{Y}_{i,j}^{m_j} \widetilde{Y}_{i,q+1}^{usk})$. Store $cred_i = (h, \sigma_i)$.

Fig. 7. Credential Issuing Algorithm.

and n_I public and private key pairs $\{isk_i, ipk_i\}_{i=1}^{n_I}$. TTP selects (x, y) as the private key and computes the verification key pk of URS signature. TTP then selects $2(q+2)$ polynomials of $t_I - 1$ degrees with coefficients in $\mathbb{Z}_p$ and uses the Shamir secret sharing scheme to computes a private key isk_i and public key ipk_i for each issuers I_i ($i \in [1, n_I]$). TTP then publishes pk and $(i, ipk_i)_{i=1}^{n_I}$, and transmits isk_i to the corresponding I_i through the secure channel.

$\mathsf{TraceKeyGen}(pp) \rightarrow (tsk_i, tpk_i)$: T_i takes pp as input and generates ElGamal private key tsk_i and public key tpk_i. T_i selects a random $tsk_i \leftarrow R\,\mathbb{Z}_p$, and computes $tpk_i = \widetilde{g}^{tsk_i}$. T_i stores tsk_i and publishs (i, tpk_i).

$\mathsf{UKeyGen}(id, pp) \rightarrow (usk, upk)$: U takes their identity id and pp as inputs, then U selects $usk \leftarrow R\,\mathbb{Z}_p$ and computes $h = \mathcal{H}_2(id)$ and public key $upk = h^{usk}$.

$\langle \mathsf{Obtain}(id,\ usk,\ upk,\ \{m_j\}_{j=1}^{q}, ipk_i, \{tpk_j\}_{j=1}^{n_T},\ pp) \leftrightarrow \mathsf{Issue}\ (isk_i, ipk_i, pp)\rangle \rightarrow cred_i$: As shown in Fig. 7, U and I_i generate a user's partial attribute-based credential $cred_i$ through the following interaction:

– U takes the following steps to generate registration information: (1) To perform blind signatures on the set $\{m_j\}_{j=1}^{q}$ of private attributes, U generates the ElGamal private key z and public key $Z = g^z$, and computes the ciphertext (α_j, β_j) for each attribute m_j ($j \in [1, q]$). (2) To aggregate t_I partial credentials from different issuers into single, each issuer must operate on the same random element $h = g^r$ where r is unknown, thus U generates $h = \mathcal{H}_2(id)$ using the collision-resistant hash function. (3) To support threshold tracing/revocation, U selects a polynomial $f_{usk}(x)$ of degree $t_T - 1$ with coefficients in $\mathbb{Z}_p$, which is used to divide usk into n_T shares $(s_1, \ldots, s_{n_T})$

$$\langle \mathsf{Show}(usk, \{m_j\}_{j=1}^{q}, \mathcal{D}, cred, pp) \leftrightarrow \mathsf{Verify}(pk, pp)\rangle \rightarrow (0/1, tok)$$

User U

Denote $\overline{\mathcal{D}} = [1,q]\backslash\mathcal{D}, \mathcal{D}' = \mathcal{D} \cup \{q+1\}, \mathbb{D} = \{m_j\}_{j\in\mathcal{D}}$.

Select random $(r,t) \xleftarrow{R} \mathbb{Z}_p$, and compute :

$\sigma_1' = h^r$,

$\sigma_2' = \sigma^r(\sigma_1')^t$,

$\widetilde{\sigma}' = \widetilde{g}^t \cdot \prod_{j\in\overline{D}} \widetilde{Y}_j^{m_j}$.

For all $j \in \mathcal{D}'$, compute $c_j = \mathcal{H}_1(\sigma_1', \sigma_2', \widetilde{\sigma}', \mathbb{D}, j)$.

Compute $\sigma_3' = \prod_{j\in D'}[Y_{q+2-j}^t \prod_{i\in\overline{D}} Y_{q+2-j+i}^{m_i}]^{c_j}$.

Compute $C = (\sigma_1')^{usk}$ and the proof Π_2 :

$\mathsf{SoK}\{usk : C = (\sigma_1')^{usk}\}$.

Set $tok = (\sigma_1', \sigma_2', \sigma_3', \widetilde{\sigma}', C, \mathbb{D}, \Pi_2)$.

$\xrightarrow{tok}$

Verifier V

Verify the proof Π_2.

For $j \in \mathcal{D}'$, compute $c_j = \mathcal{H}_1(\sigma_1', \sigma_2', \widetilde{\sigma}', \mathbb{D}, j)$.

Check $e(\sigma_1', \widetilde{X}\widetilde{\sigma}' \prod_{j\in D} \widetilde{Y}_j^{m_j})e(C, \widetilde{Y}_{q+1}) = e(\sigma_2', \widetilde{g})$ and $e(\sigma_3', \widetilde{g}) = e(\prod_{j\in D'} Y_{q+2-j}^{c_j}, \widetilde{\sigma}')$.

For each revocation information rev in $\mathcal{L}$, check $e(\sigma_1', rev) \neq e(C, \widetilde{Y}_{q+1})$.

Fig. 8. Credential Showing Algorithm.

$$\langle \mathsf{Trace}(tsk_i, tok)\rangle_{i\in\mathcal{T}} \rightarrow (id/\perp)$$

```
1: For all $id$, $\mathsf{T}_i$ reads the registration information in $\mathcal{L}$, and parse it as $(id, Z, upk, \{\alpha_j, \beta_j\}_{j=1}^{q}, \{\widetilde{C}_{1,i}, \widetilde{C}_{2,i}\}_{i=1}^{n_T}, \{D_j\}_{j=1}^{t_I-1}, \Pi_1)$, and computes $S_{id,i} = e(\sigma_1', \widetilde{C}_{2,i}\widetilde{C}_{1,i}^{-tsk_i})$.
2: $\mathsf{T}_i$ broadcasts $T_i = (id, i, S_{id,i})$ to all the other $t_I - 1$ tracers $\{\mathsf{T}_j\}_{j\in\mathcal{T}, j\neq i}$.
3: After receiving $\{T_j\}_{j\in\mathcal{T}, j\neq i}$ from other tracers, $\mathsf{T}_i$ performs the following operations:
     – for each $i \in \mathcal{T}$, compute $\lambda_i = [\prod_{j\in\mathcal{T}, j\neq i}(j)][\prod_{j\in\mathcal{T}, j\neq i}(j-i)]^{-1}$;
     – if $e(C, \widetilde{Y}_{q+1}) = \prod_{j\in\mathcal{T}} S_{id,j}^{\lambda_j}$, then return $id$.
4: If all the registration information in $\mathcal{L}$ does not satisfy the above equation, return $\perp$.
```

Fig. 9. Tracing Algorithm.

using the Shamir's secret sharing scheme. U use the ElGamal encryption scheme to encrypt each share s_i of usk. And then U computes verification values $\{D_j\}_{j=1}^{t_T-1}$ as in the Feldman verifiable secret sharing scheme [26] to verify the correctness of the private key usk sharing. (4) To prove to the issuer that all computations are correct, U needs to construct a proof of knowledge Π_1, and send registration information reg to all issuers.

– After verifying Π_1, I_i signs upk and ciphertexts $(\alpha_j, \beta_j)_{j=1}^{q}$ of private attributes, and sends the blind signature (α_i', β_i') to U. I_i stores the registration information reg in $\mathcal{L}$.

– After receiving (α_i', β_i'), using the homomorphism of the ElGamal encryption scheme, U can unblind the signature and get a partial credential $cred_i$ of private attributes.

$\mathsf{CredAgg}(\{cred_i\}_{i\in\mathcal{I}}, pp) \rightarrow cred$: The algorithm can be executed after U receives t_I partial credentials. For each $i \in \mathcal{I}$, U computes $\lambda_i = [\prod_{j\in\mathcal{I}, j\neq i}(j)][\prod_{j\in\mathcal{I}, j\neq i}(j-i)]^{-1}$, and then computes $\sigma = \prod_{i\in\mathcal{I}} \sigma_i^{\lambda_i}$. If $e(\sigma, \widetilde{g}) = e(h, \widetilde{X}\prod_{j=1}^{q} \widetilde{Y}_j^{m_j} \widetilde{Y}_{q+1}^{usk})$, U keeps $cred = (h, \sigma)$, otherwise returns $\perp$.

$\langle \mathsf{Show}(usk, \{m_j\}_{j=1}^{q}, \mathcal{D}, cred, pp) \leftrightarrow \mathsf{Verify}(pk, pp)\rangle \rightarrow (0/1, tok)$: As shown in Fig. 8, U interacts with V to show credentials anonymously. In this protocol, U is allowed to disclose a subset of attributes. First, U derives credential $cred$ as in the URS signature. In addition, U need to construct a proof of knowledge Π_2 to prove to V that he knows usk. Finally, U sends the token tok to V. If the proof and signature are verified and the user's credentials have not been revoked, then V returns 1 and stores tok, otherwise returns 0. It should be noted that it is inefficient to directly apply the formula to compute $\widetilde{\sigma}'$, because it requires $|\mathcal{D}'|(q+2-|\mathcal{D}'|)$ exponentiations, but fortunately, in [7], Sanders gives a way that only requires $2(q+1)-1$ exponentiations.

$\langle \mathsf{Trace}(tsk_i, tok)\rangle_{i\in\mathcal{T}} \rightarrow (id/\perp)$: As shown in Fig. 9, t_T tracers jointly perform identity tracing operations. To trace the id of the user who generated tok, tracers traverse the latest ledger $\mathcal{L}$, read the registration information of each user in it, and determine whether the usk hidden in tok is the same as the one restored by their t_T shares. If it is the case, the algorithm returns the corresponding id. If $\mathcal{L}$ is traversed and the id is not determined, it returns $\perp$.

$\langle \mathsf{Revoke}(tsk_i, id, pp)\rangle_{i\in\mathcal{T}} \rightarrow rev$: As shown in Fig. 10, t_T tracers jointly perform user revocation operations. For the malicious user id whose identity has been traced, each T_i $(i \in \mathcal{T})$ retrieves their registration information in $\mathcal{L}$, and decrypts the revocation information share $R_{id,i} = \widetilde{C}_{2,i}\widetilde{C}_{1,i}^{-tsk_i} = \widetilde{Y}_{q+1}^{s_i}$. The complete revocation information $rev = \prod_{j\in\mathcal{T}} R_{id,j}^{\lambda_j} = \widetilde{Y}_{q+1}^{usk}$ can be recovered by using the t shares $\{R_{id,j}\}_{j\in\mathcal{T}}$. Finally, T_i stores the revocation information rev in $\mathcal{L}$.

```
⟨Revoke$(tsk_i, id, pp)\rangle_{i\in\mathcal{T}} \to rev$
1: $\mathsf{T}_i$ reads the user $id$'s registration information in $\mathcal{L}$, and parse it as $(id, Z, upk, \{\alpha_j, \beta_j\}_{j=1}^{q}, \{\widetilde{C}_{1,i}, \widetilde{C}_{2,i}\}_{i=1}^{n_{\mathcal{T}}}$,
   $\{D_j\}_{j=1}^{t_I-1}, \Pi_1)$, and computes $R_{id,i} = \widetilde{C}_{2,i}\widetilde{C}_{1,i}^{-tsk_i}$.
2: $\mathsf{T}_i$ broadcasts $Q_i = (id, i, R_{id,i})$ to all the other $t_I - 1$ tracers $\{\mathsf{T}_j\}_{j\in\mathcal{T}, j\neq i}$.
3: After receiving $\{T_j\}_{j\in\mathcal{T}, j\neq i}$ from other tracers, $\mathsf{T}_i$ performs the following operations:
   – for each $i \in \mathcal{T}$, compute $\lambda_i = [\prod_{j\in\mathcal{T}, j\neq i}(j)][\prod_{j\in\mathcal{T}, j\neq i}(j-i)]^{-1}$;
   – output $rev = \prod_{j\in\mathcal{T}} R_{id,j}^{\lambda_j}$, and store $rev$ into $\mathcal{L} \leftarrow rev$.
```

Fig. 10. Revocation Algorithm.

TABLE II
FEATURES COMPARISON ($\surd$: SUPPORTED FEATURE; $-$: UNSUPPORTED FEATURE)

Scheme	Threshold Issuing	Threshold Tracing	Threshold Revocation	Blind Issuance	Re-randomization	Attribute Disclosure Proof	Relation Proof
CL [40], [10]	√	√	√	–	–	–	–
BBS [40], [12]	√	√	√	–	–	–	–
PS [14], [13]	√	√	–	–	–	–	–
SPS-EQ[16]	–	–	–	√	√	SPS-EQ	–
ART-Sign [39]	–	–	–	√	√	ART-Sign	–
Coconut [4]	√	–	–	√	√	ZKP	√
Our scheme	√	√	√	√	√	URS	√

C. Correctness and Security

The details of the signature of knowledge of the proposed scheme are shown in *Supplemental Material C*, available online. The correctness of the our scheme is analyzed in *Supplemental Material D*, available online. We define the following theorems to formalize that our construction from Section IV-B satisfies all the desired security guarantees defined in Section III-C. The formal proofs are given in *Supplemental Material E*, available online.

Theorem 1: Our scheme achieves anonymity if DDH assumption holds in $\mathbb{G}_1$.

Theorem 2: Our scheme achieves blindness if ElGamal encryption is IND-CPA secure.

Theorem 3: Our scheme achieves traceability if the URS is unforgeable.

Theorem 4: Our scheme achieves non-frameability if DL assumption holds in $\mathbb{G}_1$.

D. Applications

Our scheme retains all the excellent features of the Coconut and supports all the applications introduced by Sonnino et al. [4]. In *Supplementary Material A*, available online, we show two application scenarios: accountable anonymous reporting system and permissioned token system, that leverage our scheme to offer improved fine-grained authentication, traceability, and privacy properties.

V. PERFORMANCE ANALYSIS

A. Theoretical Analysis and Comparison

Table II makes a features comparison of our scheme with typical anonymous credential schemes. The CL [10], [40] and BBS [12], [40] support threshold issuing, threshold tracing and threshold revocation, PS [13], [14] schemes support threshold issuing and threshold tracing, but they are not attribute-based credentials. The schemes in [16], [39] support blind issuance and re-randomization, and they use SPS-EQ signature and ART-sign, respectively, to replace the complex zero-knowledge proof to achieve efficient attribute disclosure proof. Still, their scheme does not support relation proofs for hidden attributes and threshold functions. Coconut [4] supports threshold issuing, blind issuance, re-randomization, and relational proof. However, it does not support threshold tracing and threshold revocation; another issue of Coconut is that it relies on zero-knowledge proof to achieve attribute disclosure proof, which is not highly efficient. Our scheme achieves all the features listed in Table II. In particular, it relies on URS to achieve attribute disclosure proof more efficiently than Coconut.

Table III provides a security comparison of our scheme with other threshold anonymous credential schemes. The scheme in [40] supports anonymity, traceability, and non-frameability, but does not support blind issuance of private attributes and does not provide a formal security definition. The scheme in [14] supports anonymity, traceability, and non-frameability, and provides a formal definition of security, but does not support blind issuance of private attributes. The scheme in [4], [5] supports anonymity, blindness, and non-frameability, but it doesn't support traceability.

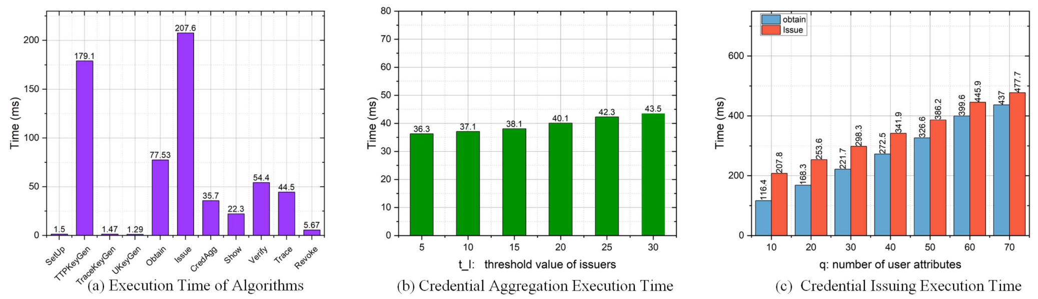
In Table IV, our scheme is compared with the Coconut [4] in terms of storage and computational complexities, where $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, $|\mathbb{Z}_p|$ are the element sizes in group $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{Z}_p$, respectively; t_{e_1}, t_{e_2}, t_p are the time cost for the exponential in group $\mathbb{G}_1$, $\mathbb{G}_2$ and pairing computations, respectively. The main overheads of these two schemes are essentially the same (constant credential size, $O(q)$ public key storage complexity

TABLE III
SECURITY COMPARISON ($\surd$: SUPPORTED FEATURE; $-$: UNSUPPORTED FEATURE)

Scheme	Formal definition	Blindness	Anonymity	Traceability	Non-Frameability
CL [40]	$-$	$-$	$\surd$	$\surd$	$\surd$
BBS [40]	$-$	$-$	$\surd$	$\surd$	$\surd$
PS [14]	$\surd$	$-$	$\surd$	$\surd$	$\surd$
Coconut [4], [5]	$\surd$	$\surd$	$\surd$	$-$	$\surd$
Our scheme	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

TABLE IV
STORAGE AND COMPUTATION COMPARISON (q/k: NUMBER OF USER/DISCLOSED ATTRIBUTES)

Scheme	Storage Overheads			Computation Overheads of Credential Showing	
	Public key Size	Credential Size	Token Size	Show Credential	Verify Credential
Coconut [4]	$(q+2)\lvert\mathbb{G}_1\rvert$	$1\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	$3\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$ $+(q+1+k)\mathbb{Z}_p$	$2(q+2)e_1$	$(q+2)e_1 + 2e_p$
Our scheme	$(3+2q)\lvert\mathbb{G}_1\rvert$ $+(q+2)\mathbb{G}_2$	$2\lvert\mathbb{G}_1\rvert$	$4\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$ $+(k+2)\mathbb{Z}_p$	$[2(q+2)+1]e_1$ $+(q-k+2)e_2$	$(k+2)e_1+$ $ke_2 + 5e_p$



(a) Execution Time of Algorithms (b) Credential Aggregation Execution Time (c) Credential Issuing Execution Time

Fig. 11. Execution Time of Algorithms.

and credential showing complexity, etc) except for the verification cost and token size. The verification cost of credentials in Coconut is $O(q)$ operations, while it is $O(k)$ operations in our scheme. The token size in Coconut is $O(q)$ elements, while it is $O(k)$ elements in our scheme. The improvements of our scheme are significant in practice where q is usually much larger than k.

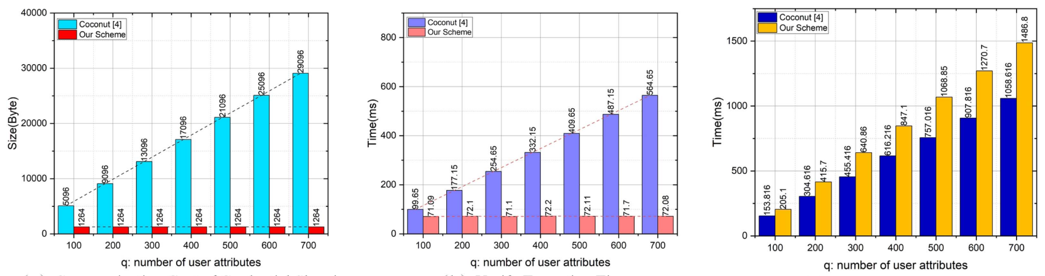
B. Experimental Analysis

In order to evaluate the performance in objective tests, we implement our scheme and measure its performance on a personal laptop with an AMD Ryzen-5 4600H with Radeon Graphics 3.00 GHz CPU, 16 GB RAM, 512 GB SSD running Ubuntu Kylin 16.04 operating system. The experiments are conducted using MIRACL[1] and Type-III pairing. We use SHA256 to implement the hash functions $\mathcal{H}_1, \mathcal{H}_2$ required by our scheme (see Figs. 7, 8). In order to accurately evaluate the computational overhead of algorithms in our scheme, we use the Barreto-Naehrig curve (BN-256) [42] to test the system's performance under AES-100 b security level.

Fig. 11(a) shows the computation costs of our scheme. We test the execution time of the scheme under $n_I = 5, t_I = 3, n_T = 5, t_T = 3, q = 10$, and $k = 3$. The computation costs of Setup, TTPKeyGen, TraceKeyGen, UKeyGen and CredAgg are 1.5 ms, 179.1 ms, 1.47 ms, 1.29 ms, 35.7 ms, respectively. For credential issuing, it costs U and I_i for 77.53 ms and 207.6 ms, respectively. For credential showing, it costs U and V for 22.3 ms and 54.4 ms, respectively. It takes 44.5 ms for T_i to check whether a user satisfies the conditions in the Trace algorithm, and it takes 5.67 ms for T_i to compute the revocation information of a user in the Revoke algorithm. Fig. 11(b) shows the computation costs of the credential aggregation algorithm when the number of issuers $n_I = 30$ and the threshold value t_I changes. It costs 36.3 ms, 37.1 ms, 38.1 ms, 40.1 ms, 42.3 ms and 43.5 ms for $t_I = 5, 10, 15, 20, 25$ and 30, respectively. In Fig. 11(c), we show the computation costs of the credential issuing algorithm. When $n_I = 5, t_I = 3, n_T = 5, t_T = 3$, and $k = 3$, the computation time increases linearly with q. Fortunately, the credential issuing algorithm is executed only once for each new user.

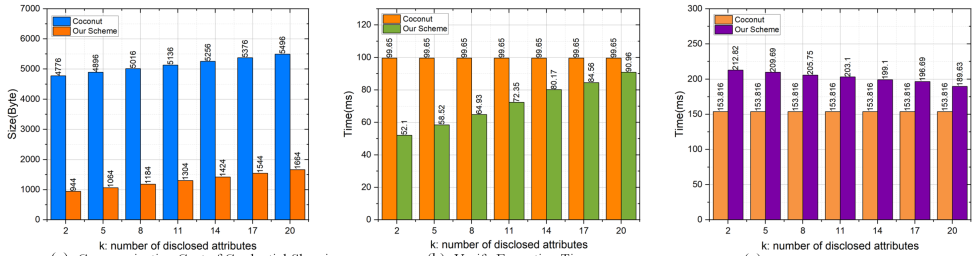
In practice, the number q of a user's attributes is usually much larger than the number k of a user's disclosed attributes. For example, the user may have $name, id, student, driver, license, social, security, number, occupation, home, address$, and many other attributes. If the user buys a student discount ticket, they only need to disclose the $student$ attribute. Therefore, we compare the performance of our scheme with Coconut [4] under the conditions of a large q and a small k. Note that the following experiment does not include the revocation checking operations, since Coconut does not support user revocation.

[1]https://github.com/miracl/



(a) Communication Cost of Credential Showing (b) Verify Execution Time (c) Show Execution Time

Fig. 12. Overheads of Credential Showing (k is kept constant).



(a) Communication Cost of Credential Showing (b) Verify Execution Time (c) Show Execution Time

Fig. 13. Overheads of Credential Showing (q is kept constant).

As shown in Fig. 12, we compare the communication and computation cost of the user (performing Show) and verifier (performing Verify) for the credential showing algorithm, where k is fixed to be 10 and q linearly increases. In Fig. 12(a), we compare the sizes of the token tok between our scheme and Coconut [4]. The communication cost of Coconut grows linearly with the number q of user's attributes. Since the size of tok in our scheme is independent of q, when k is constant, our communication consumption is constant and substantially less than Coconut. When $q = 100$ (resp. $q = 700$), the communication cost of our scheme is approximately 24.8% (resp. 4.3%) of that in Coconut. In Fig. 12(b), we compare the credential verification time between Coconut [4] and our scheme. The computation cost of Coconut grows linearly with the number q of user's attributes. Since the operations of credential verification in our scheme is independent of q, when k is constant, our computation cost is constant and substantially less than Coconut. When $q = 100$ (resp. $q = 700$), the computation cost of our scheme is approximately 71.3% (resp. 12.7%) of that in Coconut. In Fig. 12(c), we compare the credential showing time between Coconut [4] and our scheme. The computation cost of both our scheme and Coconut grows linearly with q. Although our scheme requires more computation than Coconut when showing credentials, there is no significant gap between them.

As shown in Fig. 13, we compare the communication and computation cost of the user and verifier for the credential showing algorithm, where q is fixed to be 100 and k linearly increases. In Fig. 13(a), we compare the sizes of the token tok between our scheme and Coconut [4]. The communication cost of both our scheme and Coconut grows linearly with the number k of disclosed attributes, but the size of our scheme is significantly smaller than that of Coconut. When $k = 2$ (resp. $k = 20$), the communication cost of our scheme is approximately 19.7% (resp. 30.2%) of that in Coconut. In Fig. 13(b), we compare the credential verification time between Coconut [4] and our scheme. The computation cost of our scheme grows linearly with the number k of disclosed attributes, but the time consumption in our scheme is still lower than that of Coconut. When $k = 2$ (resp. $k = 20$), the computation cost of our scheme is approximately 52.2% (resp. 91.3%) of that in Coconut. In Fig. 13(c), we compare the credential showing time between Coconut [4] and our scheme. The computation cost of our scheme decreases linearly with the number k of disclosed attributes while the computational cost of Coconut remains constant. The execution time of our scheme is higher than that of Coconut, but they are still in the same order.

The above analysis and comparison indicate that our scheme enjoys low communication and computation overheads.

C. Evaluation of Smart Contract

As shown in Fig. 14, we present the smart contract of our scheme to demonstrate the feasibility of deploying our scheme on the blockchain. The contract has the following functions: Create, AddIssuer, AddTracer, uploadReg, getReg, uploadCred, getCred, uploadToken, getToken, uploadTi, getTi, uploadID, getID, uploadQi, getQi, uploadRev and getRev. First, TTP deploys the smart contract and calls the Create function setting system parameters, including n_I, t_I, n_T, t_T, and then calls the AddIssuer and AddTracer functions to publish the public keys ($address$) of all issuers and tracers. To obtain attribute-based credentials, a new user uploads their registration information to the smart contract via the uploadReg function.

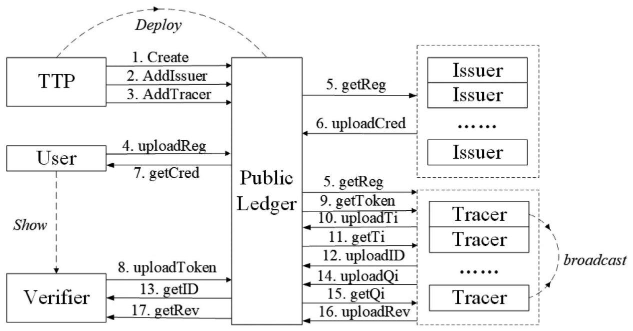


Fig. 14. Smart Contract in our Scheme.

The issuer (online) is then responsible for monitoring the smart contract and getting the user's registration request via the getReg function and uploading a partial credential via the uploadCred function if the registration request is valid. The user can obtain the aggregated credential after downloading the partial credentials issued by t issuers via the getCred function. When a verifier applies to trace the identity of the user who created an illegal token, the verifier uploads the token to the smart contract via the uploadToken function. Then the tracer is responsible for monitoring the smart contract and getting the token via the getToken function. For cooperative execution of the tracing algorithm, the tracer (online) uploads the calculated tracing information T_i to the smart contract via the uploadTi function and obtains the tracing information of other tracers via the getTi function. After the tracer gets the t shares of the tracing information and verifies them, it uploads the user's identity to the smart contract via the uploadID function, and broadcasts the tracing success message to other tracers. The verifier can get the tracing result via the getID function. To revoke the credentials of the malicious user id, the tracer uploads the calculated revocation information Q_i to the smart contract via the uploadQi function and gets revocation information from other tracers via the getQi function. After the tracer receives t shares of revocation information and verifies them, it uploads the complete revocation information of the user id to the smart contract, and broadcasts the revocation success message to other tracers. The verifier can get the updated revocation list via the getRev function.

The smart contract was implemented using *Solidity* language.[2] The algorithm of the smart contract is shown in *Supplementary Material F*, available online, where the public keys of all participants are represented by the type $address$, and the input data is represented by the type $string$.

Then, we evaluate the gas consumption of each function on the Ethereum test chain (Remix VM Merge) using the Remix.[3] For $n_I = 5, t_I = 3, n_T = 5, t_T = 3, q = 10$, and $k = 3$, we list the gas consumption and the corresponding Ether cost and US dollar cost for each function call in Table V. The gas consumption covers the transaction cost of Ethereum and the execution cost of our smart contract. The most expensive operation is the

[2]https://soliditylang.org/
[3]http://remix.ethereum.org/

TABLE V
THE GAS CONSUMPTION FOR OUR SCHEME (GASPRICE = 1 GWEI, 1 ETH = 1605 USD)

Function	GAS	ETH	USD
Create	135,691	0.000135691	0.217784055
AddIssuer	46,726	0.000046726	0.07499523
AddTracer	46,703	0.000046703	0.074958315
uploadReg	5,383,032	0.005383032	8.63976636
getReg	531,123	0.000531123	0.852452415
uploadCred	230,668	0.000230668	0.37022214
getCred	101,714	0.000101714	0.16325097
uploadToken	817,841	0.000817841	1.312634805
getToken	82,259	0.000082259	0.132025695
uploadTi	481,770	0.00048177	0.77324085
getTi	262,357	0.000262357	0.421082985
uploadID	49,414	0.000049414	0.07930947
getID	3,919	0.000003919	0.006289995
uploadQi	299,270	0.00029927	0.48032835
getQi	161,821	0.000161821	0.259722705
uploadRev	274,612	0.000274612	0.44075226
getRev	34,530	0.00003453	0.05542065

uploadReg function, which costs 8.6 dollar and fortunately only needs to be called once for each new user. Calling uploadToken costs 1.3 dollar, but this function is only called for illegal tokens. It costs less than 1 dollar for each of the remaining function calls.

VI. CONCLUSION

In this article, we proposed a threshold attribute-based anonymous credential scheme that has three significant practical consequences. First, it supports threshold tracing of user identities and threshold revocation of user credentials, making it better suited in anonymous-yet-accountable distributed scenarios. Second, it relies on the polynomial-based URS signatures for credential verification, making it more efficient. Finally, formal security proofs of our scheme are provided for anonymity, blindness, traceability, and non-frameability. However, the revocation of attributes in anonymous credentials is still a challenging issue, and we leave it as our future work.

REFERENCES

[1] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger[J]," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.
[2] C. Cachin, "Architecture of the hyperledger blockchain fabric," *Proc. Workshop Distrib. Cryptocurrencies Consensus Ledgers*, vol. 310, no. 4, pp. 1–4, 2016.
[3] M. Al-Bassam et al., "Chainspace: A sharded smart contracts platform[J]," 2017, *arXiv: 1708.03778*.
[4] A. Sonnino et al., "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers," 2018, *arXiv:1802.07344*.
[5] A. Rial and A. M. Piotrowska, "Security analysis of coconut, an attribute-based credential scheme with threshold issuance," *Cryptol. ePrint Arch.*, vol. 2022, 2022, Art. no. 11.
[6] IRMA, "Privacy by design foundation," 2016. Accessed: Jun. 5, 2023. [Online]. Available: https://privacybydesign.foundation/attribute-index/en/
[7] O. Sanders, "Improving revocation for group signature with redactable signature[J]," *IACR Cryptol. ePrint Arch.*, vol. 2020, 2020, Art. no. 856.
[8] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *Proc. 5th Int. Conf. Secur. Cryptogr. Netw.*, R. De Prisco and M. Yung, Eds., Springer, Heidelberg, Sep. 2006, pp. 111–125.
[9] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, S. Cimato, C. Galdi, and G. Persiano Eds., Springer, Heidelberg, Sep. 2003, pp. 268–289.

[10] J. Camenisch and A. Lysyanskaya ., “Signature schemes and anonymous credentials from bilinear maps,” in *Proc. 24th Annu. Int. Cryptol. Conf.*, M. Franklin Ed., Springer, Heidelberg, Aug. 2004, pp. 56–72.
[11] D. Boneh and X. Boyen, “Short signatures without random oracles and the SDH assumption in bilinear groups,” *J. Cryptol.*, vol. 21, no. 2, pp. 149–177, Apr. 2008.
[12] D. Boneh, X. Boyen, and H. Shacham, “Short group signatures,” in *Proc. Annu. Int. Cryptol. Conf.*, M. Franklin Ed., Springer, Heidelberg, Aug. 2004, pp. 41–55.
[13] D. Pointcheval and O. Sanders, “Short randomizable signatures ,” *Proc. Cryptogr. Track RSA Conf.*, Springer, Cham, 2016, pp. 111–126.
[14] J. Camenisch, M. Drijvers, A. Lehmann, G. Neven, and P. Towa, “Short threshold dynamic group signatures,” in *Proc. Secur. Cryptogr. Netw.: 12th Int. Conf.*, Amalfi, SA, Italy, Sep. 14–16, 2020, pp. 401–423.
[15] G. Fuchsbauer, C. Hanser, and D. Slamanig, “Practical round-optimal blind signatures in the standard model,” in *Proc. Adv. Cryptol.–CRYPTO: 35th Ann. Cryptol. Conf.*, Santa Barbara, CA, USA, Aug. 16–20, 2015, pp. 233–253.
[16] G. Fuchsbauer, C. Hanser, and D. Slamanig, “Structure-preserving signatures on equivalence classes and constant-size anonymous credentials[J],” *J. Cryptol.*, vol. 32 no. 2, pp. 498–546, 2019.
[17] E. C. Crites and A. Lysyanskaya, “Delegatable anonymous credentials from mercurial signatures,” in *Proc. Topics Cryptol.–CT-RSA: Cryptogr. Track RSA Conf.*, San Francisco, CA, USA, Mar. 4–8, 2019, pp. 535–555.
[18] E. C. Crites and A. Lysyanskaya, “Mercurial signatures for variable-length messages,” *Proc. Privacy Enhancing Technol.*, vol. 2021, no. 4, pp. 441–463, 2021.
[19] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss, “Composable and modular anonymous credentials: Definitions and practical constructions,” in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, Berlin, Heidelberg, 2015, pp. 262–288.
[20] J. Groth and A. Sahai, “Efficient non-interactive proof systems for bilinear groups,” in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, Springer, Berlin, Heidelberg, 2008, pp. 415–432.
[21] O. Sanders, “Efficient redactable signature and application to anonymous credentials,” in *Proc. IACR Int. Conf. Public-Key Cryptogr.*, Springer, Cham, 2020, pp. 628–656.
[22] T. Elgamal, “Public key cryptosystem and a signature scheme based on discrete logarithms[J],” *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.
[23] A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[24] R. Gennaro et al., “Secure distributed key generation for discrete-log based cryptosystems,” in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, Springer, Berlin, Heidelberg, 1999, pp. 295–310.
[25] A. Kate, Y. Huang, and I. Goldberg, “Distributed key generation in the wild[J],” *IACR Cryptol. ePrint Arch.*, vol. 2012, 2012, Art. no. 377.
[26] P. Feldman, “A practical scheme for non-interactive verifiable secret sharing,” in *Proc. 28th Annu. Symp. Found. Comput. Sci.*, 1987, pp. 427–438.
[27] S. D. Galbraith, K. G. Paterson, and N. P. Smart, “Pairings for cryptographers[J],” *Discrete Appl. Math.*, vol. 156, no. 16, pp. 3113–3121, 2008.
[28] J. Katz, *Digital Signatures*, Berlin, Germany: Springer Science and Business Media, 2010.
[29] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Proc. Conf. Theory Appl. Cryptogr. Techn.*, Springer, Berlin, Heidelberg, 1986, pp. 186–194.
[30] M. Chase and A. Lysyanskaya, “On signatures of knowledge,” in *Proc. Annu. Int. Cryptol. Conf.*, Springer, Berlin, Heidelberg, 2006, pp. 78–96.
[31] J. Camenisch and M. Stadler, “Efficient group signature schemes for large groups,” in *Proc. Annu. Int. Cryptol. Conf.*, Springer, 1997, pp. 410–424.
[32] H. Wang, D. He, Z. Liu, and R. Guo, “Blockchain-based anonymous reporting scheme with anonymous rewarding[J],” *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1514–1524, Nov. 2020.
[33] E. B. Sasson et al., “Zerocash: Decentralized anonymous payments from bitcoin,” in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 459–474.
[34] S. F. Sun et al., “RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero,” in *Proc. Eur. Symp. Res. Comput. Secur.*, Springer, Cham, 2017, pp. 456–474.
[35] E. Androulaki et al., “Privacy-preserving auditable token payments in a permissioned blockchain system,” in *Proc. 2nd ACM Conf. Adv. Financial Technol.*, 2020, pp. 255–267.
[36] C. Diaz et al., “Privacy preserving electronic petitions[J],” *Identity Inf. Soc.*, vol. 1, no. 1, pp. 203–219, 2008.
[37] C. Garman, M. Green, and I. Miers, “Decentralized anonymous credentials,” *Cryptol. ePrint Arch.*, vol. 2014, 2013, Art. no. 622.
[38] R. Yang et al., “Decentralized blacklistable anonymous credentials with reputation[J],” *Comput. Secur.*, vol. 85, pp. 353–371, 2019.
[39] C. Hébant and D. Pointcheval, “Traceable constant-size multi-authority credentials,” in *Proc. Secur. Cryptogr. Netw.: 13th Int. Conf.*, Amalfi, SA, Italy, Sep. 12–14, 2022, pp. 411–434.
[40] R. Gennaro, S. Goldfeder, and B. Ithurburn, “Fully distributed group signatures,” 2019. Accessed: Jun. 5, 2023. [Online]. Available: https://www.orbs.com/assets/docs/white-papers/Crypto_Group_signatures-2.pdf
[41] M. Abdalla, C. Namprempre, and G. Neven, “On the (Im) possibility of blind message authentication codes,” in *Proc. Cryptogr. Track RSA Conf.*, Springer, Berlin, Heidelberg, 2006, pp. 262–279.
[42] J. Fan, F. Vercauteren, and I. Verbauwhede, “Faster $\mathbb{f}_p$-Arithmetic for cryptographic pairings on barreto-naehrig curves,” in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, Springer, Berlin, Heidelberg, 2009, pp. 240–253.
[43] EU Whistleblower, “Protection directive,” 2019. Accessed: Jun. 5, 2023. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32019L1937

Rui Shi is working toward the PhD degree under the supervison of prof. Huamin Feng and prof. Yang Yang with the school of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China. He is also a research engineer with Beijing Electronic Science and Technology Institute. His research interests are in the area of privacy protection and cryptography.

Huamin Feng is currently a professor with Beijing Electronic Science and Technology Institute and Beijing University of Posts and Telecommunications. His research interests include information security and cyberspace security.

Yang Yang (Member, IEEE) received the BSc degree from Xidian University, Xi’an, China, in 2006, and the PhD degrees from Xidian University, China, in 2011. She is a full professor with the College of Computer Science and Big Data, Fuzhou University. She is also a research fellow with the School of Computing and Information System, Singapore Management University. Her research interests are in the area of information security and privacy protection. She has published more than 60 papers in *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Industrial Informatics*, etc.

Feng Yuan received the PhD degrees from Xidian University, Xi’an, China, in 2010. Now, he is a assistant researcher with Beijing Electronic Science and Technology Institute. His research interests are in the area of cryptography and information security.

Yingjiu Li received the PhD degree from George Mason University, in 2003. He had been a faculty member with Singapore Management University from 2003 to 2019. Now, He is ripple professor with the Department of Computer and Information Science, University of Oregon. His research interests include IoT security and privacy, mobile security, and data security and privacy. He has published more than 130 papers in Cybersecurity, and co-authored two academic books.

Hwee Hwa Pang received the BSc (first class honors) and MS degrees from the National University of Singapore, in 1989 and 1991, respectively, and the PhD degree from the University of Wisconsin-Madison, in 1994, all in computer science. He is a professor with the School of Computing and Information Systems, Singapore Management University. His current research interests include database management systems, data security, and information retrieval.

Robert H. Deng (Fellow, IEEE) is AXA chair professor of cybersecurity with the School of Computing and Information Systems, Singapore Management University. His research interests include data security, network and system security. He has served/is serving on the editorial boards of many international journals in security, such as *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, etc.