AC-TIR: Fast Anonymous Credentials with Threshold Issuance and Revocation

Anonymous Author(s)

Abstract

Anonymous credentials (AC) offer privacy for applications like authentication and authorization. They allow users to obtain attribute-based credentials from an issuer and disclose some necessary attributes for privacy-preserving authentications and authorizations. The security of AC requires the credentials to be unforgeable and accountable to prevent users from misusing their issued credentials. AC also desires to support a decentralized system with different issuers, some of which might be dishonest. To date, no AC schemes simultaneously satisfy our defined privacy (unlinkability and untraceability) and security (linkability, traceability and unforgeability) in decentralized settings.

We present an anonymous credential scheme with threshold issuance and revocation (AC-TIR) that supports our defined privacy and security. Threshold issuance and revocation allow a threshold number of issuers to grant and revoke credentials. Privacy ensures that the users cannot be traced, and their credential showings cannot be linked before being revoked. Security guarantees the opposite side of privacy, and the revoked credentials remain unforgeable. In particular, we propose a new privacy-preserving signature primitive to construct our AC-TIR. We provide benchmarks to show that our AC-TIR scheme runs fast for large-scale systems.

Keywords

Anonymous Credentials, Threshold Revocation, Privacy, Security.

ACM Reference Format:

Anonymous Author(s). 2024. AC-TIR: Fast Anonymous Credentials with Threshold Issuance and Revocation. In . ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Anonymous credential (AC), introduced by Chaum [18], is a privacy-friendly and user-centric solution for authentication and authorization. In AC schemes, a certificate authority (CA) issues credentials to users, and each credential includes a signature from CA on a set of attributes (e.g., a user's name, age, and gender). Users can then show that they hold valid credentials by selectively disclosing a subset of attributes associated with the credentials (i.e., selective disclosure). ACs have various applications. For example, anonymous web browsing such as Privacy Pass [22], the private groups in the Signal messaging app [17], and the PrivateStats to anonymously collect client data from WhatsApp [42]. ACs can also be used to design direct anonymous attestation (DAA) schemes in TPM 2.0, which have been implemented in millions of chips [12].

An AC holds certain properties. The fundamental one is privacy-preserving. Specifically, a verifier can acknowledge that the attributes involved in a credential satisfy a statement, but the verifier (even colluding with the CA that issued the credential) cannot identify the user or link multiple showings of the same credential. Another essential property is security, which includes accountability and unforgeability. The AC scheme must ensure that authorized parties (e.g., CAs) can revoke the credentials that may be lost, stolen, or associated with malicious users. The revoked credentials should stay unforgeable. Threshold support is also necessary when an AC is used in a decentralized system. Threshold support considers issuance and revocation. Threshold issuance allows a threshold number of CAs to issue partial credentials for a user, who then derives a valid credential from those partial credentials. Similarly, revocation could be executed by a threshold number of CAs. Threshold ACs have wide applications, including Chainspace [41], (offline) anonymous payment systems [29, 37], decentralized finance (DeFi) [35], credential coalescing [25], and decentralized identity [44]. Furthermore, NIST is putting efforts into standardizing threshold cryptography [1].

Motivation. The state-of-the-art threshold AC schemes include Coconut [41] and its follow-up work Coconut′ [36], but they lack accountability. Moreover, Coconut lacks a formal security analysis and its unforgeability cannot be proven, thus it cannot be considered to be secure. Coconut′ has doubled the CA's public-key size to ensure a valid security analysis, thus less practical. Other relevant works are called ZEBRA [35] and IhMA [32]. Briefly, ZEBRA provides threshold revocation, but lacks threshold issuance and selective disclosure. IhMA supports compact-size credentials from multiple CAs, but lacks user revocation. The detailed comparison refers to Section 7. Our motivation is to propose an AC scheme that incorporates all the considered properties efficiently.

Our Results. In this work, we introduce a secure and fast anonymous credential scheme with threshold issuance and revocation, which we refer to as an AC-TIR scheme. Our AC-TIR construction mainly relies on a new modification of the Pointcheval-Sanders (PS) signature [34] scheme Σ_{PS} and a threshold public-key encryption (TPKE). We start with introducing Σ_{PS}, which is of independent interest. Our proposed Σ_{PS} signature is based on the well-known randomizable PS signature, a Linear encryption scheme [9], and zero-knowledge (ZK) proofs. Our Σ_{PS} signature offers unlinkability and untraceability, which are not considered in the original PS signature. Briefly, we modify the original PS signature to ensure unlinkability and use the Linear encryption to realize untraceability. We assume the signed messages can be (un)disclosed, in which the undisclosed messages are proven in zero-knowledge. Unlinkability

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

means no one can link the randomized PS signatures to the disclosed messages, and untraceability means no one can trace the randomized PS signatures to the undisclosed messages. Our Σ_{PS} signature significantly differs from the existing modified PS signatures used in the anonymous credentials [21, 29, 32, 36, 37, 41]. The PS-type signature in [41] is insecure as no unforgeability analysis is given. The PS-type signature in [21, 29, 36, 37] extends the CA's public key, and the PS-type signature in [32] requires an extra pairing operation for verification compared to the original PS signature. Our unforgeable Σ_{PS} signature avoids extending public keys and adding extra pairings.

Moving to our proposed AC-TIR scheme, our privacy model considers unlinkability and untraceability. Specifically, we allow multiple CAs to issue partial credentials (each of which is a Σ_{PS} signature) to a user for an issuance. We consider that any CAs (representing as verifiers) cannot link multiple showings of a valid credential (i.e., an aggregated version of a set of Σ_{PS} signatures) and trace those showings to a credential issuance, *unless* the user holding a valid credential was revoked. Our security model considers accountability (i.e., linkability and traceability) and unforgeability. It is non-trivial to realize such accountability because TPKE alone cannot guarantee it. A *simple* solution is to let a user create a TPKE ciphertext encrypting a unique identifier (or called a revocation tag), a threshold number of CAs decrypt the ciphertext and recover the tag. Also, the user may create a ZK proof that the ciphertext is well formed. However, the problem is that the user can always choose an arbitrary value to create a TPKE ciphertext and a ZK proof to avoid such accountability.

Our technique is to use revocation tags to associate the proposed Σ_{PS} scheme with the TPKE scheme. We assume each user owns a unique revocation tag m_0. For credential issuance, a threshold number of CAs create Σ_{PS} signatures on the same signed message that is derived from m_0. In this case, the user holding a valid Σ_{PS} signature cannot replace m_0 with an arbitrary value due to the signature's unforgeability. During the prove protocol, the user creates a randomized Σ_{PS} signature, a ZK proof and a TPKE ciphertext associated with m_0. Any verifier shall verify the validity of the randomized Σ_{PS} signature and the ZK proof, and perform an additional check against a public revocation list. The user's privacy is preserved if the revocation check fails, i.e., the revocation tag m_0 does not belong to the revocation list. To ensure accountability, a threshold number of CAs can recover m_0 by decrypting a TPKE ciphertext and include m_0 to the revocation list, which follows the above simple solution. As a result, anyone can link the multiple showings of the same credential and the CAs can trace a credential's multiple showings to its issuance. Overall, the key insight of our accountability is that the message-signature pairs in our proposed Σ_{PS} scheme involves revocation tags, *otherwise* the revoked user may create ZK proofs and TPKE ciphertexts associated with arbitrary values and make the revocation check always fail.

Last, we provide the detailed implementation to show the efficiency of our proposed AC-TIR scheme as well as the state-of-the-art schemes Coconut and Coconut′. Our implementation details the performance of these schemes by considering various numbers of attributes and authorities to meet different AC system requirements. Our scheme performs well compared to the non-accountability-based Coconut and Coconut′. For example, when considering 10 authorities in an AC system and 100 attributes to be encoded in a credential, the threshold issuance takes $\approx$ 1.2s, and the prove protocol and the verify protocol take $\approx$ 0.8s and 0.5s respectively. This is fast enough for most AC systems. The detailed evaluation is given in Section 6.

2 Overview of AC-TIR

AC-TIR is an anonymous credential scheme, supporting threshold credential issuance of attributes, privacy-preserving credential shows, threshold credential revocations and the ability to disclose a subset of attributes. We assume n authorities (Auths) share a common public key, and each holding a share of the private key. The threshold number of Auths is assumed to be t, where $t \in [1, n]$. We assume a user requests a single credential. We can easily extend it to a scenario where multiple users request multiple credentials, and we consider this scenario in our definition and models.

In our AC-TIR, a user sends a request command to t Auths. In response to the request command, each authority independently responds with an issue command to create a partial credential. The user aggregates a threshold number t of partial credentials to form a valid credential. We allow a set of attributes to be encoded into a credential, and each attribute set contains a unique revocation tag representing the user. Later, the user with a valid credential can selectively disclose the encoded attributes, and create a proof to show s/he knows the undisclosed attributes encoded in the credential. A verifier shall check the proof is valid against the common public key and the user is not revoked against a revocation list called verifier-local revocation (VLR). We allow a threshold number t of Auths to revoke users by adding their revocation tags to VLR, and such VLR can be stored in a public platform such as blockchain. Our AC-TIR has the following properties:

- **Threshold authorities.** Only a threshold number of authorities are required to issue partial credentials to a user. The user can derive a valid credential encoding a set of attributes from those partial credentials. Similarly, a threshold number of authorities can revoke the issued credential via a public revocation list VLR.
- **Privacy and Security.** A user is required to show the possession of a valid credential in a privacy-preserving manner, meaning that anyone, even a subset of authorities (i.e., the subset size is $\leq t$), cannot learn the undisclosed attributes, trace the origin of the credential, and link multiple shows of the credential. If a threshold number of authorities include the user's revocation tag to VLR, the user's privacy is lost but the revoked credential remains unforgeable.
- **Efficiency.** The size of partial/valid credential is independent of the number of authorities and the number of attributes encoded in the credentials. The execution time of AC-TIR is required to be fast. Our implementation confirmed the efficiency of our scheme, compared to the non-accountability-based schemes [36, 41].

3 Preliminaries and Building Blocks

We present the mathematical assumptions and our proposed PS-type signature.

3.1 Complexity Assumptions

Let $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ be three cyclic groups of prime order p, and $\hat{e} : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$ is a type-3 bilinear map.

Definition 3.1. Let h be a generator in $\mathbb{H}$. For (h^x, h^y) where $(x, y) \in \mathbb{Z}_p^2$, this assumption *Ass* defines an oracle $O(m)$ on input $m \in \mathbb{Z}_p$ that chooses a random $g \in \mathbb{G}$, outputs a pair (g, g^{x+my}). Given (h, h^x, h^y) and unlimited access to this oracle, no adversary can efficiently compute such a pair, with $g \neq 1_{\mathbb{G}}$, for a new $m^* \in \mathbb{Z}_p$, not asked to O.

This assumption *Ass* refers to the Assumption 2 proposed in [34], which is a weaker assumption than Assumption 1 (also proposed in [34]) that gives g^y to the adversary.

Definition 3.2. Given $(a, b, z) \in \mathbb{Z}_p^3$ and $(u, v, w) \in \mathbb{G}^3$, where (u, v, w) are uniformly distributed in $\mathbb{G}$, we define the advantage of an adversary $\mathcal{A}$ in solving a decisional linear (DLIN) problem as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{DLIN}}(\lambda) = \Pr[\mathcal{A}(u, v, w, u^a, v^b, w^{a+b}) = \mathcal{A}(u, v, w, u^a, v^b, w^z)]$$

The DLIN problem is hard if $\mathsf{Adv}_{\mathcal{A}}^{DLIN}(\lambda)$ is negligible in λ.

3.2 The Proposed PS Signature

The proposed Σ_{PS} signature should satisfy unforgeability, unlinkability and untraceability. We construct it from the original PS signature [34], a Linear encryption scheme [9] and a non-interactive zero-knowledge (NIZK) proof. Unlike the original PS signature, we assume a signer blindly signs messages chosen by a user, and we use Linear encryption to realize this blind signing for untraceability. Briefly, the user "encrypts" the signed messages while the signer signs the ciphertext. Then, the user "decrypts" the ciphertext and obtains a PS signature on the blindly signed messages. Later, the user "randomizes" the PS signature and proves the ownership of the message-signature pair for unlinkability. The user computes the NIZK proofs of knowing secret witnesses (e.g., the undisclosed messages and randomness). The resulting Σ_{PS} signature is unforgeable; no third parties can forge the signer's PS signatures and the user's NIZK proofs. Below, we present step-by-step guidance explaining our designs.

3.2.1 The Unlinkable PS signature. Our first task is to build a PS-type signature that satisfies unlinkability. The PS signature [34] is a randomizable signature scheme, which includes a single-message version and a multiple-message version. We sketch a single-message setting below.

- **Setup**: It takes a security parameter λ as input, outputs $pp = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \hat{e})$.
- **KeyGen**: It takes pp as input, outputs a key pair $\mathsf{sk} = (x, y) \in \mathbb{Z}_p^2$, $\mathsf{pk} = (h, \alpha, \beta) = (h, h^x, h^y) \in \mathbb{H}^3$.
- **Sign**: It selects a random $g \in \mathbb{G}$ and a message $m \in \mathbb{Z}_p$, outputs $\sigma = (g, g^{x+my})$.
- **Verify**: It verifies $g \neq 1_{\mathbb{G}}$ and $\hat{e}(g^{x+my}, h) = \hat{e}(g, \alpha \cdot \beta^m)$.

The original PS signature achieves unforgeability, which is proven under the assumption *Ass*. But, the original PS signature cannot ensure unlinkability (i.e., CPA-full-anonymity [9]), meaning that the attackers cannot decide which message has been used in the generation of a PS signature σ given two "disclosed" messages (m_0, m_1). Now, we analyze the PS signature's unlinkability as follows.

$$\textcircled{1}: \quad \hat{e}(g^{x+m_b y}, h) = \hat{e}(g, \alpha \cdot \beta^{m_{b'}}).$$
$$\textcircled{2}: \quad \hat{e}(g^{(x+m_b y)r'}, h) = \hat{e}(g^{r'}, \alpha \cdot \beta^{m_{b'}}).$$
$$\textcircled{3}: \quad \hat{e}(g^{x+m_b y} \cdot g^r, h) = \hat{e}(g, \alpha \cdot \beta^{m_b} \cdot h^r).$$
$$\textcircled{4}: \quad \hat{e}(g^{x+(m_b - m_{b'})y} \cdot g^r, h) = \hat{e}(g, \alpha \cdot \beta^{(m_b - m_{b'})} \cdot h^r).$$
$$\textcircled{5}: \quad \hat{e}(g^{(x+m_b y)r'} \cdot g^{r'r}, h) = \hat{e}(g^{r'}, \alpha \cdot \beta^{m_b} \cdot h^r).$$

From equation ①, the attackers can choose $m_{b'}$ and decide the message m_b trivially, where $b \in [0, 1]$ is chosen by a simulator, and $b' \in [0, 1]$ is chosen by attackers. There are two approaches to make PS signature unlinkable. The first approach is to compute $\sigma = (g^{r'}, g^{(x+m_b y)\cdot r'})$, where $r' \in \mathbb{Z}_p$. Clearly, the approach is not working as attackers can still break unlinkability (i.e., equation ②). The second approach is to compute $\sigma = (g, g^{x+m_b y} \cdot g^r)$ and $\mathsf{pk} = (h, \alpha, \beta, \underline{\alpha \cdot \beta^{m_b} \cdot h^r})$, where $r \in \mathbb{Z}_p$. The signature σ is valid against pk via equation ③. The attackers, even if holding $\mathsf{sk} = (x, y)$, cannot decide the message m_b. This is because, the pairing holds in case of $b = b'$ and $b \neq b'$ via equation ④. To ensure the proposed Σ_{PS} scheme is unlinkable, we compute a signature $\sigma = (g^{r'}, g^{(x+m_b y)\cdot r'} \cdot g^{r'r})$ and a NIZK proof on randomness r. The signature σ is also valid against $\mathsf{pk} = (h, \alpha, \beta, \underline{\alpha \cdot \beta^{m_b} \cdot h^r})$ via equation ⑤.

Comparison to [36, 41]: The PS signature in the Coconut [41] cannot ensure unlinkability. This is because their PS signature is in the form of $(g^{(x+m_b y)r'}, g^{r'})$. The component $g^{(x+m_b y)r'}$ reveals the message m_b based on the analysis above. The original PS signature supports randomization in the sense that, the user can choose a random group element $g' = g^{r'}$ and create a randomized signature $\sigma = (g', g'^{x+my})$ on a message m, where $r' \in \mathbb{Z}_p$. Fortunately, the Coconut′ scheme [36] addressed this privacy concern based on equation ⑤. We observe that the PS signature used in Coconut′ ensures unlinkability based on our analysis. They also introduced "unconditional privacy", but the privacy model is not given. In this work, we give the explicit definition of unlinkability (refer to our privacy model in Section 4.2) and provide a rigorous proof of our proposed Σ_{PS} scheme (see Appendix A).

3.2.2 The Untraceable PS signature. Our second task is to build a PS-type signature that satisfies untraceability. Specifically, we build a blind PS signature from a linear encryption scheme [9]. We focus on the blinded signing algorithm, which is an interactive process between a user and a signer described in Figure 1.

We show the detailed unblinding process below. Note that the linear encryption requires $w = u^{d_1} = v^{d_2}$.

$$\begin{aligned}
\frac{\tilde{W}}{\tilde{U}^{d_1} \cdot \tilde{V}^{d_2}} &= \frac{W^y \cdot g^x \cdot w^{r_1 + r_2}}{(U^y \cdot u^{r_1})^{d_1} \cdot (V^y \cdot v^{r_2})^{d_2}} \\
&= \frac{(g^m \cdot w^{s_1+s_2})^y \cdot g^x \cdot w^{r_1+r_2}}{(u^{s_1 y} \cdot u^{r_1})^{d_1} \cdot (v^{s_2 y} \cdot v^{r_2})^{d_2}} \\
&= \frac{(g^m \cdot w^{s_1+s_2})^y \cdot g^x \cdot w^{r_1+r_2}}{w^{s_1 y} \cdot w^{r_1} \cdot w^{s_2 y} \cdot w^{r_2}} = g^{x+my}
\end{aligned}$$

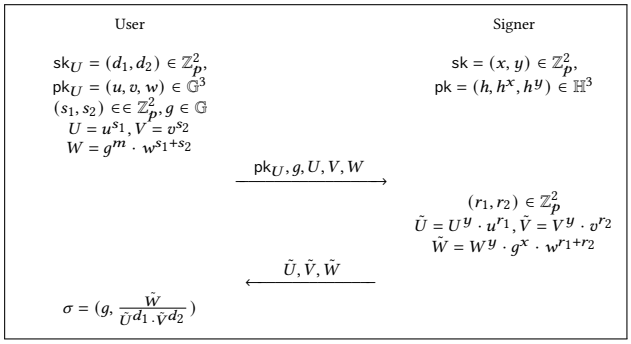

Figure 1: Blind Signing.

After unblinding, the user randomizes the signature σ, while anyone can verify such randomizable signature against pk. During the interactive signing, the user creates a NIZK proof on witnesses (m, d_1, d_2, s_1, s_2), and the signer verifies it before signing. We omit the NIZK proof in Figure 1 for simplicity.

Comparison to [36, 41]: The PS signature used in the Coconut lacks formal security analysis. Specifically, the unforgeablity of the ElGamal-based blind PS signature was not given [41]. The Coconut′ scheme [36] was proven secure in the UC model. In particular, Coconut′ extended the issuer's public-key to prove the unforgeability of the commitment-based blind PS signature. The size of public keys is two times linear to the maximum number of messages that needs to be signed. The blind signing of our proposed Σ_{PS} scheme is based on a linear encryption scheme, instead of the commitment scheme. Our intention is to prove its unforgeability without doubling the issuer's public keys.

3.2.3 Our Σ_{PS} signature. We present the proposed Σ_{PS} scheme in a single-message setting. We consider three entities: a user, a signer and a verifier. We rely on NIZK to prove the knowledge of the undisclosed message and secret randomness. The security analysis of the proposed Σ_{PS} is deferred to Appendix A.

- The setup and key generation algorithms are the same as the original PS signature.
- **Sign**: This is an interactive process between the user and the signer. The detailed blind signing process refers to Figure 1 in Section 3.2.2.
  - Step 1: The user chooses a key pair $(\mathsf{sk}_U, \mathsf{pk}_U)$, randomness (s_1, s_2), a group element $g \in \mathbb{G}$ and a message $m \in \mathbb{Z}_p$, and computes (U, V, W). The user also computes a NIZK proof π_s on $(m, \mathsf{sk}_U, s_1, s_2)$ and sends $(\mathsf{pk}_U, g, U, V, W, \pi_s)$ to the signer.
  - Step 2: If π_s is valid, the signer chooses randomness (r_1, r_2), computes $(\tilde{U}, \tilde{V}, \tilde{W})$ and sends them to the user.
  - Step 3: The user creates a valid PS signature $\sigma = (g, g^{x+my})$ using the secret key sk_U (i.e., blind factor).
- **Prove** & **Verify**: This is an interactive process between the user and the verifier. The user computes an unlinkable PS signature $(g' = g^{r'}, s' = g'^{x+my} \cdot g'^{r})$, a public key $\mathsf{pk} = (h, \alpha, \beta, \alpha \cdot \beta^m \cdot h^r)$, a NIZK proof π_v on (m, r), and publishes $(g', s', \mathsf{pk}, \pi_v)$, where $(r, r') \in \mathbb{Z}_p^2$, the verifier checks $g' \neq 1_{\mathbb{G}}$, $\hat{e}(g'^{x+my} \cdot g'^{r}, h) = \hat{e}(g', \alpha \cdot \beta^m \cdot h^r)$ and π_v is valid. The unlinkable technique refers to equation ⑤ described in Section 3.2.1.

4 Definition and Models

We present the definition and the models for an AC-TIR scheme. We consider a generic case where users request at most ℓ credentials from n authorities, each credential encodes at most q attributes and each attribute set includes a unique revocation tag m_0. Also, we consider a user requests a credential only, thus we define three indexes (i, j, l), where $i \in [1, n]$, $j \in [0, q]$ and $l \in [1, \ell]$.

4.1 Definition

The anonymous credential with threshold issuance and revocation (AC-TIR) scheme is defined as follows. The messages (resp. signatures) will be called attributes (resp. credentials) to comply with the terminology of AC schemes.

- **Setup:** Anyone (e.g., a verifier) takes a security parameter λ as input, outputs parameters pp and a revocation list VLR (verifier-local revocation). The parameters pp is implicit to all the following algorithms. The setup is trust-free.
- **KeyGen:** A trusted authority takes a parameter n and a threshold $t \in [1, n]$ as input, outputs a public key pk and n key pairs $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}$, where $i \in [1, n]$. The index i indicates the i-th authority with key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$. The key generation can be performed in a distributed fashion [27].
- **IssueCred:** It is an interactive protocol between users and t independent authorities. A user indexed with $l \in [1, \ell]$ obtains a partial credential σ_i encoding a set of attributes $(m_0, m_1, \cdots, m_q)$ satisfying a statement ϕ, i.e., $\phi(m_0, m_1, \cdots, m_q) = 1$. The attribute m_0 is a unique revocation tag involved in the l-th credential issuance, and t authorities do not need to interactive each other to coordinate issuance.
- **AggCred:** A user takes a threshold number t of partial credentials $\{\sigma_i\}$ as input, outputs an aggregated credential σ encoding $(m_0, m_1, \cdots, m_q)$ and satisfying $\phi(m_0, m_1, \cdots, m_q) = 1$.
- **ProveCred:** A user takes the public key pk, a credential σ encoding the attributes $(m_0, m_1, \cdots, m_q)$ and a statement ϕ' as input, outputs a proof Θ and the statement ϕ'.
- **VerifyCred:** A verifier takes the public key pk, a proof Θ, a statement ϕ' and VLR as input, outputs *true* if: 1) the attributes $(m_0, m_1, \cdots, m_q)$ associated with the proof Θ satisfy the statement ϕ' under pk, i.e., $\phi'(m_0, m_1, \cdots, m_q) = 1$; 2) the attribute m_0 has not been revoked, i.e., $m_0 \notin$ VLR; and 3) the proof Θ is generated by **ProveCred**. Otherwise, outputs *false*, meaning that either Θ is not a valid proof or the credential σ is revoked, i.e., $m_0 \in$ VLR.
- **RevokeCred:** It is an interactive protocol between t authorities. It takes the public key pk, a proof Θ, a statement ϕ' and VLR as input, outputs a revocation tag m_0 and adds it to VLR.

Implicit Tracing Algorithm. An AC-TIR scheme has an *implicit* tracing algorithm, meaning that authorities with a tracing key can trace a proof Θ to an execution of credential issuance. This trace process states that, if some revocation tags such as m_0 are recorded in the VLR via **RevokeCred**, then: 1) multiple proofs associated with the same m_0 (or the same credential σ) are linked; and 2) the l-th credential issuance is traced. First, this tracing algorithm requires some revocation tags to be included in VLR, i.e., $VLR = \{m_0\}$, which functions as a tracing key. Second, this tracing algorithm

considers at most ℓ credential issuance executions, and each credential issuance execution is associated with an aggregated credential σ. Formally, given a valid proof of a credential Θ, a user possessing some revocation tags can decide which credential issuance is associated with Θ using the following steps.

- For each $l \in [1, \ell]$, run the **VerifyCred** algorithm on (Θ, ϕ') with revocation list $VLR = \{m_0\}$.
- Output the index of the first credential issuance when the **VerifyCred** algorithm outputs $false$. Output $fail$ if the **VerifyCred** algorithm works properly for all ℓ crdentail issuance executions.

A secure AC-TIR scheme must satisfy three requirements: correctness, privacy and security. Correctness requires that

$\textbf{VerifyCred}(\Theta, \phi', VLR) \leftarrow true : \phi'(m_0, m_1, \cdots, m_q) = 1 \wedge m_0 \notin VLR \wedge (\Theta, \phi') \leftarrow \textbf{ProveCred}(\sigma, m_0, m_1, \cdots, m_q, \phi')$.

4.2 Security Models

Our privacy model captures unlinkability and untraceability. That is, given two distinct attributes (m_0, m_1), the authorities cannot link the attributes with any proof Θ during the credential showing or any credential σ during the credential issuance. Our privacy model allows the adversary to "control" less than t authorities, i.e., the adversary can access $\{\mathsf{sk}_i\}$ where $i \in [1, t)$. Our security model captures linkabilty, traceability and unforgeability. We allow the adversary to access a public revocation list which functions as a tracing key, and this revocation list is maintained by t authorities. We also consider two forging scenarios. The first scenario is to allow the adversary to "control" n authorities. The second scenario is to allow the adversary to "control" the user and less than t authorities. The "control" on the user side means the adversary can access the user's unique revocation tag m_0 and internal randomness which was used in the l-th **IssueCred** protocol. The unforgeability is trivially broken if the adversary "control" the user and more than t authorities.

Privacy. The adversary cannot learn anything about the undisclosed attribute m except that m satisfies a statement, or link multiple executions of **ProveCred**, or link any execution of **ProveCred** with an execution of **IssueCred**. We assume VLR is an append-only set. The privacy experiment between a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ is shown as follows.

- Setup. $\mathcal{S}$ runs **Setup** algorithm to obtain pp and VLR. Then, $\mathcal{S}$ runs **KeyGen** algorithm to obtain key pairs $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}$ for n authorities. Eventually, $\mathcal{S}$ returns all public keys to $\mathcal{A}$, publishes VLR and a threshold $t \in [1, n]$.
- Training. $\mathcal{A}$ can issue the following queries.
  - Issue. If $\mathcal{A}$ queries an attribute m, a statement ϕ and an index i (i.e., i-th authority), $\mathcal{S}$ returns a partial credential $\sigma_i \leftarrow \textbf{IssueCred}(\cdots)$ encoding attribute m to $\mathcal{A}$, where $\phi(m) = 1$.
  - Prove. If $\mathcal{A}$ queries an attribute m, a credential σ, a public key pk and a statement ϕ', $\mathcal{S}$ returns a proof $\Theta \leftarrow \textbf{ProveCred}(\cdots)$ and the statement ϕ' to $\mathcal{A}$, where $\phi'(m) = 1$.
  - Corrupt. If $\mathcal{A}$ queries a secret key of i-th authority, $\mathcal{S}$ returns sk_i to $\mathcal{A}$. The restriction is that $\mathcal{A}$ can neither corrupt more than t authorities, nor corrupt any users.
  - Revoke. If $\mathcal{A}$ queries an attribute m, $\mathcal{S}$ includes m to VLR.
- Challenge. $\mathcal{A}$ submits two challenge attributes (m_0^*, m_1^*), and a statement ϕ^* to $\mathcal{S}$. $\mathcal{S}$ chooses a random bit b, returns a credential $\sigma_b^* \leftarrow \textbf{IssueCred}(\cdots)$ and a proof $\Theta_b^* \leftarrow \textbf{ProveCred}(\cdots)$ to $\mathcal{A}$. $\mathcal{S}$ aborts if either m_0^* or m_1^* does not satisfy ϕ^*, or the challenge attributes exist in VLR. Finally, $\mathcal{A}$ outputs a bit b', its guess of b. $\mathcal{A}$ wins if $b' = b$.

We define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{Pri}}(\lambda) = |\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Pri}}(\lambda) \rightarrow 1] - 1/2|.$$

Definition 4.1. An AC-TIR scheme is privacy-preserving if for any probabilistic polynomial-time PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{Pri}}(\lambda)$ is negligible in λ.

Security. The adversary's goal is to forge a proof that cannot be traced to one of the attributes in her chosen revocation set VLR^* via the implicit tracing algorithm. Each attribute involved in the VLR^* represents an execution of credential issuance between a user and t authorities. The security experiment between a PPT adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ is shown as follows.

- Setup. $\mathcal{S}$ runs **Setup** algorithm to obtain pp and VLR. Then, $\mathcal{S}$ runs **KeyGen** algorithm to obtain key pairs $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}$ for n authorities. Eventually, $\mathcal{S}$ returns all public keys to $\mathcal{A}$. $\mathcal{S}$ sets $U \leftarrow \emptyset$ and a threshold $t \in [1, n]$, and publishes them.
- Training. $\mathcal{A}$ can issue Issue, Prove queries, while $\mathcal{S}$ answers these queries using the same method described in the privacy experiment. We consider the following scenarios to deal with Corrupt and Revoke queries.
  - If $\mathcal{A}$ issues Corrupt queries toward authorities, $\mathcal{S}$ answers these queries honestly. If $\mathcal{A}$ issues a Revoke query on an attribute m_0, $\mathcal{S}$ updates U by including m_0.
  - If $\mathcal{A}$ issues Corrupt query towards a user indexed with $l \in [1, \ell]$, $\mathcal{S}$ aborts if $\mathcal{A}$ queried Corrupt oracle towards authorities beyond t times. Otherwise, $\mathcal{S}$ returns the user's revocation tag m_0 and the internal randomness involved in the l-th credential issuance. $\mathcal{S}$ updates U by including m_0.
- Challenge. At some time, $\mathcal{A}$ outputs (Θ^*, ϕ^*) and a revocation list VLR^*. $\mathcal{A}$ wins if the following conditions hold.
  - The tuple $(\Theta^*, \phi^*, VLR^*)$ is $true$ via **VerifyCred**.
  - The proof Θ^* traces to some attributes such as m^* outside of the chosen set $U \backslash VLR^*$, or the tracing algorithm outputs $fail$.
  - σ^*/Θ^* is non-trivial, i.e., $\mathcal{A}$ did not obtain a valid σ^*/Θ^* by querying Issue/Prove oracle on some attributes m^*.

We define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{Sec}}(\lambda) = |\Pr[\mathcal{A}\ wins]|.$$

Definition 4.2. An AC-TIR scheme is secure if for any PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{Sec}}(\lambda)$ is negligible in λ.

5 The Proposed Scheme

The proposed AC-TIR consists of the following building blocks.

- A commitment scheme that guarantees *correctness*, *hiding* and *binding* properties, such as the Pedersen commitment scheme under DL assumption [33].
- A non-interactive zero-knowledge (NIZK) proof that satisfies *correctness*, *zero-knowledge*, and *soundness*. We require a simulation-sound extractability [24] in this work.

- A threshold public-key encryption scheme (TPKE) that satisfies IND-CPA security, such as threshold version of ElGamal encryption scheme [26], which is based on Shamir's SSS [40].
- A BLS signature scheme with existential unforgeability under chosen message attack (EUF-CMA) security [10].
- The proposed Σ_{PS} signature.
  We refer the reader to the listed schemes for explanations on concrete constructions such as Pedersen commitment, (threshold) ElGamal encryption and BLS signature, and the security notions like hiding, binding, zero-knowledge and simulation-sound extractability. We do not consider CCA-secure threshold ElGamal or Cramer-Shoup cryptosystems [16, 26] as CPA security is sufficient in this work. Also, we use Shamir's secret sharing scheme (SSS) [40] to thresholdize the proposed Σ_{PS} signature.

We provide the overview of the proposed AC-TIR scheme. The parties in the scheme include a user, n authorities ($\mathcal{V}_1, \cdots \mathcal{V}_n$) and a verifier. First, an authority creates a key pair and distributes the secret key shares to n authorities via secure channels. We use Shamir's SSS to realize secret share and recover. Second, a user with a set of attributes sends a request to t authorities. The request includes a NIZK proof π_s on secret witnesses and an application-specific statement ϕ, where the witnesses include the attributes $\{m_0, m_1, \cdots, m_q\}$, and the internal randomness. We use the standard DL-based sigma protocol to realize π_s, ensuring the attributes satisfy ϕ. We use BLS signature to compute a deterministic value $\tilde{g} = \mathsf{H}(c_m)$ to ensure t authorities with their key shares sign the same request. Note that c_m is a commitment derived from $(m_0, m_1, \cdots, m_q)$. Once the user receives enough partial Σ_{PS} signatures from t authorities, s/he can create a valid credential through aggregation (i.e., an aggregated Σ_{PS} signature). Third, the user creates a new statement ϕ', a new NIZK proof π_v, a proof Θ and a threshold ElGamal ciphertext (c_1, c_2) on h^{m_0}, to show the ownership of the valid credential to a verifier. Note that h denotes a group generator. Last, the verifier should check the proof Θ is valid and the revocation tag m_0 is not revoked via VLR. One technical insight is that the user cannot replace m_0 with random $m_0' \neq m_0$ to avoid accountability, because m_0 was signed by authorities and our proposed Σ_{PS} signature is proven unforgeable. The VLR-based threshold revocation allows t authorities to decrypt the threshold ElGamal ciphertext (c_1, c_2) involved in the proof Θ and add h^{m_0} to VLR. We present the detailed construction below.

Setup: This algorithm can be run by a verifier and it involves a security parameter λ. The verifier performs the following

- Let g, h be generators of $\mathbb{G}, \mathbb{H}$, and $\hat{\mathsf{e}} : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$.
- Compute q generators $g_0, g_1, \cdots g_q$ of $\mathbb{G}$, where q is the number of attributes.
- Set two hash functions $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}$, $\mathsf{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, where p is a prime number order.
- Set a verifier-local revocation list $VLR \leftarrow \emptyset$.
- Output $\mathsf{pp} = (p, q, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \hat{\mathsf{e}}, g, h, g_0, g_1, \cdots, g_q, \mathsf{H}_1, \mathsf{H}_2, VLR)$.

KeyGen. This algorithm can be run by a trusted authority and it involves input (pp, t, n). pp are the system parameters, $t \in [1, n]$ is the threshold, and n is the number of authorities. The authority performs the following

- Choose q+1 polynomials $(v, w_0, w_1, \cdots, w_q)$ of degree t-1 with random coefficients in $\mathbb{Z}_p$.
- Set a secret key $\mathsf{sk} = (x, y_0, y_1, \cdots, y_q) \leftarrow (v(0), w_0(0), w_1(0), \cdots, w_q(0))$.
- For $i = [1, n]$, set the secret key of each authority $\mathcal{V}_i$ as $\mathsf{sk}_i = (x_i, y_{i,0}, y_{i,1}, \cdots, y_{i,q}) \leftarrow (v(i), w_0(i), w_1(i), \cdots, w_q(i))$.
- For $i = [1, n]$, set the public key of each authority $\mathcal{V}_i$ as $\mathsf{pk}_i = (\alpha_i, \beta_{i,0}, \beta_{i,1}, \cdots, \beta_{i,q}) \leftarrow (h^{x_i}, h^{y_{i,0}}, h^{y_{i,1}}, \cdots, h^{y_{i,q}})$.
- Output the public key $\mathsf{pk} = (\alpha, \beta_0, \beta_1, \cdots, \beta_q) \leftarrow (h^x, h^{y_0}, h^{y_1}, \cdots, h^{y_q})$.

As a result, each authority $\mathcal{V}_i$ holds a key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$.

IssueCred. This interactive protocol between a user and an authoroty $\mathcal{V}_i$ involves three steps. Step 1, a user chooses a set of attributes $(m_0, m_1, \cdots, m_q)$ that to be signed by $\mathcal{V}_i$, and choose a statement ϕ that to be proven about the attribute set. The user performs the following

- Pick $(d_1, d_2) \in \mathbb{Z}_p^2$, compute (u, v, w) such that $w = u^{d_1} = v^{d_2}$, and set $\mathsf{sk}_U = (d_1, d_2), \mathsf{pk}_U = (u, v, w)$, .
- Pick random $o \in \mathbb{Z}_p$, and compute a commitment $c_m = g^o \prod_{j=0}^{q} g_j^{m_j}$.
- Compute $\tilde{g} = \mathsf{H}_1(c_m)$, and encryptions of each attribute. For $j \in [0, q]$, pick $(s_{j,1}, s_{j,2}) \in \mathbb{Z}_p^2$ and compute $c_j = (U_j, V_j, W_j) = (u^{s_{j,1}}, v^{s_{j,2}}, \tilde{g}^{m_j} \cdot w^{s_{j,1}+s_{j,2}})$.
- Compute a NIZK argument of knowledge π_s for the relation

$$
\begin{aligned}
\pi_s \quad = \quad & \mathrm{NIZK}\{(d_1, d_2, m_0, m_1, \cdots, m_q, o, s_{0,1}, s_{0,2}, \cdots, s_{q,1}, s_{q,2}) : \\
& u^{d_1} \wedge v^{d_2} \wedge \underline{g^{m_0}} \wedge c_m = g^o \prod_{j=0}^{q} g_j^{m_j} \\
& \wedge \{u^{s_{j,1}} \wedge v^{s_{j,2}} \wedge \tilde{g}^{m_j} \cdot w^{s_{j,1}+s_{j,2}}\}_{\forall j \in [0,q]} \\
& \wedge \phi(m_0, m_1, \cdots, m_q) = 1\}
\end{aligned}
$$

- Output $(d_1, d_2, \Lambda = (\mathsf{pk}_U, c_m, c_0, c_1, \cdots, c_q, \tilde{g}, \pi_s), \phi)$.

The NIZK on m_0 in π_s is shown here. The user chooses a randomness r_0, computes $c_0 = \mathsf{H}_2(g^{r_0})$ and $s_0 = r_0 - m_0 \cdot c_0$. The authority verifies $c_0 \stackrel{?}{=} \mathsf{H}_2(g^{m_0 \cdot c_0} \cdot g^{s_0})$, given (c_0, g, g^{m_0}, s_0). This method also applies to other secret witnesses.

Step 2, the user sends (Λ, ϕ) to t authorities. Each authority $\mathcal{V}_i$ performs the following

- Parse Λ as $(\mathsf{pk}_U, c_m, c_0, c_1, \cdots, c_q, \tilde{g}, \pi_s)$.
- Compute $\tilde{g}^\star = \mathsf{H}_1(c_m)$. Abort if $\tilde{g}^\star \neq \tilde{g}$.
- Verify π_s. Abort if the proof is invalid.
- For $j \in [0, q]$, parse c_j as (U_j, V_j, W_j) and pk_U as (u, v, w). Pick $(r_{j,1}, r_{j,2}) \in \mathbb{Z}_p^2$, and compute $\tilde{c}_i = (\tilde{U}_i, \tilde{V}_i, \tilde{W}_i) = (\prod_{j=0}^{q} U_j^{y_{i,j}} \cdot u^{r_{j,1}}, \prod_{j=0}^{q} V_j^{y_{i,j}} \cdot v^{r_{j,2}}, \tilde{g}^{x_i} \cdot \prod_{j=0}^{q} W_j^{y_{i,j}} \cdot w^{r_{j,1}+r_{j,2}})$.
- Output the blinded PS signature share $\tilde{\sigma}_i = (\tilde{g}, \tilde{c}_i)$.

Step 3, each authority $\mathcal{V}_i$ sends $\tilde{\sigma}_i$ to the user. For all the signature shares received, the user performs the following

- Parse $\tilde{\sigma}_i$ as $(\tilde{g}, \tilde{c}_i)$, where $\tilde{c}_i = (\tilde{U}_i, \tilde{V}_i, \tilde{W}_i)$.
- Compute $\sigma_i = (\tilde{g}, s_i) = (\tilde{g}, \frac{\tilde{W}_i}{\tilde{U}_i^{d_1} \cdot \tilde{V}_i^{d_2}}) = (\tilde{g}, \tilde{g}^{x_i + \Sigma_{j=0}^{q} m_j \cdot y_{i,j}})$.
- Parse pk_i as $\alpha_i, \beta_{i,0}, \beta_{i,1}, \cdots, \beta_{i,q}$ and verify the signature share. Abort if any of the equation $\hat{\mathsf{e}}(\tilde{g}, \alpha_i \prod_{j=0}^{q} \beta_{i,j}^{m_j}) = \hat{\mathsf{e}}(s_i, h)$ does not hold.
- Output the unblinded PS signature share σ_i.

AggCred. This algorithm runs by the user and it involves input $(\mathsf{pp}, \{\sigma_i\}_{i\in\mathbb{S}}, \mathsf{pk}, m_0, m_1, \cdots, m_q)$. $\mathbb{S}$ denotes the set of indices of authorities that provided signature shares, $\{\sigma_i\}_{i\in\mathbb{S}}$ are the PS signature shares output by t executions of **IssueCred**. The user performs the following

- For all $i \in \mathbb{S}$, parse σ_i as $(\tilde{g}, s_i)$.
- For all $i \in \mathbb{S}$, evaluate at 0 the Lagrange basis polynomials

$$l_i = [\prod_{j\in\mathbb{S}, j\neq i} (0-j)][\prod_{j\in\mathbb{S}, j\neq i} (i-j)]^{-1} \mod p$$

- Compute the signature $\sigma = (\tilde{g}, s) = (\tilde{g}, \prod_{i\in\mathbb{S}} s_i^{l_i}) = (\tilde{g}, \tilde{g}^{x+\Sigma_{j=0}^{q} m_j \cdot y_j})$. The value $x = \sum x_i \cdot l_i$, where l_i is the Lagrange coefficient for the i-th share and $i \in [1, t]$. This calculation also applies to $y_0, y_1, \cdots, y_q$.
- Parse pk as $(\alpha, \beta_0, \beta_1, \cdots, \beta_q)$ and verify the signature σ. Abort if the equation $\hat{e}(\tilde{g}, \alpha \prod_{j=0}^{q} \beta_j^{m_j}) = \hat{e}(s, h)$ does not hold.

ProveCred. This algorithm runs by the user and it involves input $(\mathsf{pp}, \mathsf{pk}, m_0, m_1, \cdots, m_q, \sigma, \phi')$. The user performs the following

- Parse σ as $(\tilde{g}, s)$, and parse pk as $(\alpha, \beta_0, \beta_1, \cdots, \beta_q)$.
- Pick random $(r, r') \in \mathbb{Z}_p^2$, and compute $\tilde{g}' = \tilde{g}^{r'}$, $s' = \tilde{g}^{(x+\Sigma_{j=0}^{q} m_j \cdot y_j + r) r'}$, and $\kappa = \alpha \prod_{j=0}^{q} \beta_j^{m_j} h^r$.
- Pick random $k_0 \in \mathbb{Z}_p$, and compute a threshold ElGamal ciphertext $(c_1, c_2) = (h^{k_0}, \beta_0^{k_0} \cdot h^{m_0})$.
- Compute a NIZK argument of knowledge π_v

$$\pi_v = \mathrm{NIZK}\{(k_0, m_0, m_1, \cdots, m_q, r) : \underline{c_1 = h^{k_0}} \wedge \underline{c_2 = \beta_0^{k_0} \cdot h^{m_0}} \wedge \underline{(\tilde{g}')^{m_0}} \wedge \kappa = \alpha \prod_{j=0}^{q} \beta_j^{m_j} h^r \wedge \phi'(m_0, m_1, \cdots, m_q) = 1\}$$

- Set $\Theta = (\kappa, \tilde{g}', s', \pi_v)$, and output (Θ, ϕ').

The NIZK on m_0 in π_v is shown here. The user chooses a randomness r_0, computes $c_0 = \mathsf{H}_2(\tilde{g}'^{r_0})$ and $s_0 = r_0 - m_0 \cdot c_0$. Anyone verifies $c_0 \stackrel{?}{=} \mathsf{H}_2(\tilde{g}'^{m_0 \cdot c_0} \cdot \tilde{g}'^{s_0})$, given $(c_0, \tilde{g}', \tilde{g}'^{m_0}, s_0)$. This method also applies to other secret witnesses.

VerifyCred. This algorithm runs by a verifier and it involves input $(\mathsf{pp}, \mathsf{pk}, \Theta, \phi')$ and VLR. The verifier performs the following

- Parse Θ as $(\kappa, \tilde{g}', s', \pi_v)$.
- *NIZK Check.* Output $false$ if the proof π_v is invalid.
- *Signature Check.* Output $false$ if $\tilde{g}' = 1$ or $\hat{e}(\tilde{g}', \kappa) = \hat{e}(s', h)$ does not hold.
- *Revocation Check.* Output $false$ if $\hat{e}((\tilde{g}')^{m_0}, h) = \hat{e}(\tilde{g}', \underline{h^{m_0}})$ holds, for any $h^{m_0} \in VLR$. If no element of VLR is encoded in π_v, the user identified with m_0 has not been revoked.
- Output $true$.

RevokeCred. This algorithm runs by t authorities and it involves $(\mathsf{pp}, \mathsf{pk}, \Theta, \phi')$ and VLR.

- For all $i \in \mathbb{S}$, the i-th authority computes $c_1^{y_{i,0}}$ and generates a NIZK proof on $y_{i,0}$.
- For all $i \in \mathbb{S}$, evaluate at 0 the Lagrange basis polynomials

$$l_i = [\prod_{j\in\mathbb{S}, j\neq i} (0-j)][\prod_{j\in\mathbb{S}, j\neq i} (i-j)]^{-1} \mod p$$

- Compute the revocation tag $h^{m_0} = c_2 / \prod_{i\in\mathbb{S}} c_1^{y_{i,0} \cdot l_i}$. This step is done by one of t authorities (i.e., a combiner).
- Include h^{m_0} to the revocation list VLR. This step is done by the combiner.

Correctness. We provide the correctness of the claimed accountability (i.e., linkability and traceability). Threshold revocation is straightforward as t authorities can decrypt the threshold ElGamal ciphertext (c_1, c_2) involved in a proof Θ and recover a revocation tag in the form of h^{m_0}. The value m_0 is private to anyone except the user. First, the VLR-based revocation defeats unlinkability. The verifier can easily link the previously-generated proofs from the same revocation tag m_0 such as $(\Theta_1', \Theta_2') = (\tilde{g_1}', s_1', \cdots, \tilde{g_2}', s_2', \cdots)$ via the pairing operation in the *Revocation Check*.

$$\hat{e}((\tilde{g_1}')^{m_0}, h) = \hat{e}(\tilde{g_1}', \underline{h^{m_0}})$$
$$\hat{e}((\tilde{g_2}')^{m_0}, h) = \hat{e}(\tilde{g_2}', \underline{h^{m_0}})$$

Second, the implicit tracing algorithm defeats untraceability. Suppose we consider a user obtains two credentials (σ_1, σ_2) encoding two distinct revocation tags $(m_{0,1}, m_{0,2})$ from t authorities via **IssueCred** and **AggCred** algorithms. Every t authorities can trace the specific credential issuance via the following pairing operations.

$$\hat{e}(g^{m_{0,1}}, h) = \hat{e}(g, \underline{h^{m_{0,1}}})$$
$$\hat{e}(g^{m_{0,2}}, h) = \hat{e}(g, \underline{h^{m_{0,2}}})$$

The values $(g^{m_{0,1}}, g^{m_{0,2}})$ can be found at NIZK proofs in the **IssueCred** protocol, please refer to the underlined part g^{m_0} in π_s. Therefore, the tracing algorithm allows the authorities to trace a proof Θ to a credential issuance. The condition is that the corresponding revocation tag is included in the VLR. Otherwise, no one, even less than t authorities collude together cannot link multiple proofs, and trace a proof to a credential issuance. This completes the correctness of our accountability. Our privacy and security analysis are deferred to Appendix B.

Remarks on t authorities. Our proposed scheme considers a scenario where t authorities issue and revoke credentials, and these authorities share the same secret key sk. One may consider another scenario where two independent sets of authorities are involved in the scheme: t_1 number of issuing authorities and $t_2 \neq t_1$ number of revocation authorities. This scenario might enable us to relax the proposed privacy and security models. But, it sacrifices performance. Shamir's SSS should be executed twice regarding two secret keys: one is shared by issuing authorities, and another is shared by revocation authorities. To balance two scenarios, one can replace Shamir's SSS with dynamic proactive secret sharing (DPSS) protocols for stronger security and better usability [31]. In this new scenario, the membership of authorities and the threshold values can be periodically changed, and the single secret key sk remains fixed. We leave its concrete construction as a future work. Another remark is that the threshold t is fixed during setup as we do not consider dynamic TPKE [23].

6 Implementation and Evaluation

We implement our scheme, Coconut and Coconut′ schemes. Our implementation runs in Python and uses the Charm 0.5 framework [3]. Our implementation supports MNT224 curve and BN254 curve for type-3 pairings. We only present our implementation based on MNT224 curve as the efficiency gaps between our considered AC schemes remain consistent. The execution time below is measured on a PC with a 3.59 GHz AMD Ryzen 5 3600 6-Core Processor and 16GB RAM. The source code is available at Github [2].

Our implementation chooses the number of attributes and authorities within [10, 100] respectively, and sets the threshold as $t = 6$. From Figure 2, we can see that: 1) The setup time between our scheme and Coconut is nearly identical, and the setup of Coconut′ runs slower due to the generation of the double-sized pubic keys. 2) The efficiency between our scheme and Coconut/Coconut′ in terms of prove and verify is very close. Our prove and verify run slightly slower than Coconut/Coconut′ because our scheme incurs extra TPKE ciphertext generation and revocation check. This reflects our design goal: adding minimal overhead to Coconut to ensure accountability. 3) The efficiency of threshold issuance is various according to the number of attributes and authorities. When we consider [10, 100] attributes and choose 10 authorities such as Figure 2 (b), our threshold issuance runs slower than Coconut′. However, when we increase the number of authorities such as 20 authorities in Figure 2 (f), our threshold issuance runs faster than Coconut′. In particular, the threshold issuance in Coconut′ performs badly when the number of authority increases. An interesting observation is that the (insecure) Coconut always performs better than Coconut′ and our scheme. This observation is against the claim made by Coconut′ that their scheme is slightly more "efficient" than Coconut. We provide further details in our evaluation. To this end, our implementation provides a useful benchmark to some ACs in Table 1 with threshold issuance and revocation.

6.1 Evaluation

Comparison to Coconut and Coconut′. We treat Coconut and Coconut′ similarly due to their minor differences. We do not consider communication overhead between user and verifier. During a credential proof, the Coconut/Coconut′ requires the user to generate a randomized PS siganture and a NIZK. Our scheme requires the user to additionally generate a threshold ElGamal ciphertext. The verifier at Coconut/Coconut′ and our scheme should verify the randomized PS signature and the NIZK proof. Different from Coconut/Coconut′, our scheme requires the verifier to check the revocation tag encoded in the proof has not been revoked via VLR.

Our scheme incurs a minimal *extra* overhead compared to Coconut and Coconut′. First, we include a revocation tag m_0 in the credential generation (i.e., $\tilde{g}$ includes m_0), and allow the user to provide a NIZK proof of m_0. Also, we replace the ElGamal encryption in Coconut and the commitment scheme in Coconut′ with linear encryption scheme without trivially extending the public keys. Second, we let the user include a threshold ElGamal ciphertext to encrypt the revocation tag h^{m_0}. The user should create a NIZK of ciphertext to guarantee that the encrypted message h^{m_0} is consistent with the NIZK proof of m_0 used in π_v. The verifier should verify the NIZK proof, and check the revocation tag encoded in the credential has not been revoked via a "search" to VLR. The complexity of such search is linear to the number of revoked users, but like most VLR-based solutions [9], the number of revoked users is a small value.

We specifically compare our scheme against Coconut′ because Coconut′ is proven secure and is claimed more "efficient" than Coconut. We focus on key generation and threshold issuance algorithms. First, our implementation shows that the execution time of key generation algorithm between our scheme and Coconut′ is close. Both requires $\approx$ 0.4s when considering 100 attributes and 10 authorities. The key difference is that the storage overhead of Coconut′ is two times linear to the number of attributes. Second, our implementation shows that the threshold issuance in Coconut′ is not efficient. The threshold issuance involves three steps: user prepares blind sign (step 1), authorities blindly sign (step 2) and user unblinds (step 3). The step 1 and 2 of Coconut′ runs slightly faster than our scheme. But the step 3 of Coconut′ runs much slower. This is because the computational complexity at user side in Coconut′ depends on the number of attributes and authorities. For storage overhead, the user in Coconut′ should store a set of blind factors during step 3, while our scheme requires the user to store a constant number of blind factors. To conclude, our scheme's threshold issuance runs faster than Coconut′ and slower than (insecure) Coconut when considering 100 attributes and 20+ authorities. These efficiency gaps will increase when considering larger-scale use cases. Moreover, the prove and verify protocol's execution time gaps between our scheme and non-accountability-based Coconut and Coconut′ remain consistent. We provide real applications of our proposed AC-TIR in Appendix C.

7 Related Work

Anonymous Credentials. The research on anonymous credential (AC) is diverse, such as limited usage [5], revoked [14], audited [8], delegated [20], updated [7], on/off-chain [43], issuer-hiding [11] and distributed ACs [41].

Sonnino et al. [41] introduced a distributed AC called Coconut. Coconut is mainly based on a threshold blind PS signature scheme. Briefly, a user collects and aggregates a threshold number of partial PS signatures, and proves the ownership of the aggregated PS signature in a privacy-preserving manner. Later, Rial and Piotrowska [36] provided a rigorous security analysis on Coconut using the universal composability (UC) framework [15]. They pointed out certain security concerns on Coconut and provided a modified scheme called Coconut′. Specifically, they proposed a commitment-based blind PS signature as the ElGamal encryption-based blind PS signature in Coconut lacks unforgeability analysis. However, their commitment-based blind PS signature doubled the CA's public-key to achieve unforgeability. They also used it to construct (offline) anonymous e-cash system Π_{EC} [37]. The limitation of this commitment-based blind PS signature is that the public-key size is *two times* linear to the number of attributes/authorities in Coconut′/Π_{EC}. Our Σ_{PS} scheme relies on linear encryption [9], which achieves unforgeability and avoids doubling public keys. Our AC-TIR scheme offers accountability, which is not considered in [36, 37, 41].

Damgård et al. balanced privacy and accountability in blockchain-based systems [21], such as ZCash and Monero. We call it $\Pi_{\mathbf{id-layer}}$.

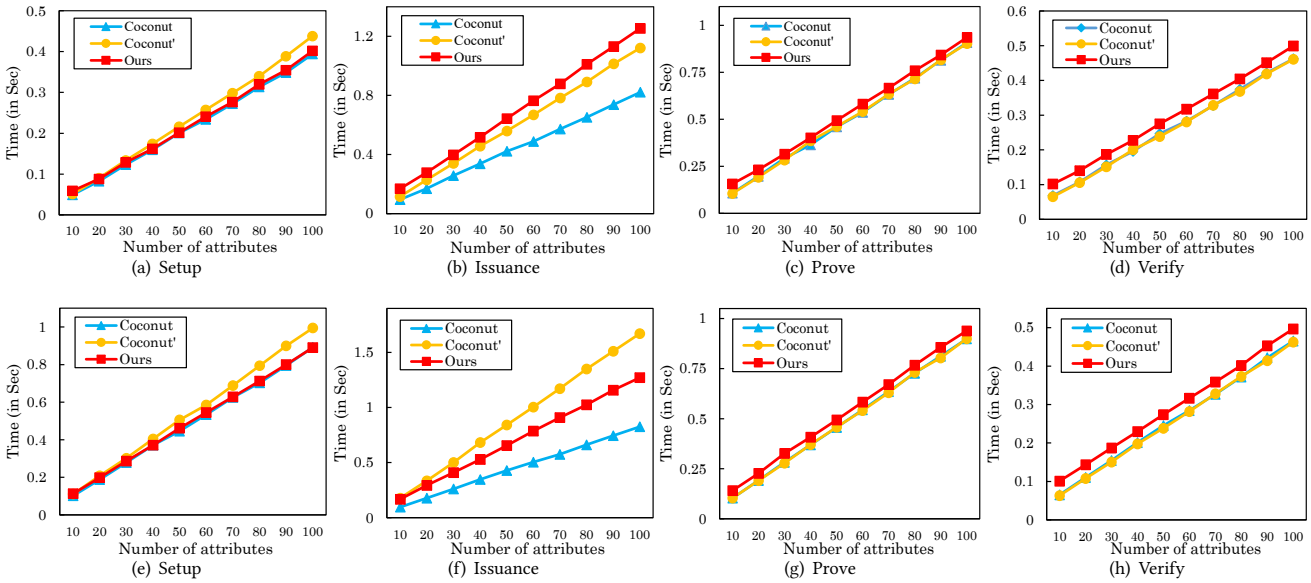

Figure 2: Execution time for AC schemes. The subfigures above chooses ten authorities, while the subfigures below chooses twenty authorities.

Table 1: The property-wise comparison of various AC schemes.

	Coconut[41]	CanDID[30]	$\Pi_{\mathbf{id-layer}}$[21]	EL PASSO[45]	Coconut′[36]	ZEBRA[35]	PEReDi[29]	IhMA[32]	tBBS+[25]	Hades[44]	RTACO[6]	IHAC [38]	Ours
Selective Disclose	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓
Threshold	✓	×	✓	✓	✓	✓	✓	×	✓	✓	✓	×	✓
Blind Issue	✓	✓	✓	✓	✓	×	✓	×	✓	×	✓	×	✓
Issuer-Hiding	×	×	×	×	×	×	×	✓	×	×	×	✓	×
DID	×	✓	×	×	×	×	×	×	×	✓	×	×	×
Audit	×	×	×	×	×	✓	✓	×	×	✓	×	✓	×
Trace	×	✓	✓	✓	×	×	✓	×	×	✓	✓	×	✓
Revoke	×	✓	✓	×	×	✓	✓	×	×	✓	✓	×	✓
Security Analysis	×	Game	UC	×	UC	Simulation	UC	Game	UC	×	UC	GGM	Game

The main idea is to introduce an identity layer to realize: 1) privacy for its users, such that anyone cannot learn the owner of any account in the system; 2) accountability (i.e., revoke and trace), such that some authorized parties (i.e., identity provider and anonymity revoker) can access the transaction history of a given user. Technically, their system relies on a set of cryptographic schemes, including commitment, PRF, blind PS signature, TPKE, and zk-SNARKs. Their system is proven in the UC framework. We list two differences only. First, their blind PS signature follows the definition of two-round blind signature schemes [39], which ensures unforgeability, blindness and *simulatability*. The third property is required in the UC framework. Their instantiated PS signature doubles the signer's public-key to ensure unforgeability. Our Σ_{PS} scheme avoids doubling public keys, and our Σ_{PS} scheme satisfies unforgeability, blindness and unlinkability. Second, their system requires an identity provider and multiple anonymity revokers to run a MPC protocol to revoke a user and trace all accounts belongs to that user. Our AC-TIR's accountability requires no MPC.

Zhang et al. introduced EL PASSO [45], a privacy-preserving single sign-on scheme based on anonymous credentials. First, their acccountability allows a threshold number of authorities to decrypt an ElGamal ciphertext and identify a user's identity in a database managed by an (trusted) identity manager. We treat such accountability as trace only, as removing the user's identity from the identity database is not supported. Our AC-TIR's accountability relies on a revocation list VLR to record the revoked users. Second, their blind PS signature is based on commitment scheme, and their scheme has no formal security analysis.

Kiayias et al. [29] introduced a fully privacy-preserving and regualted central bank digital currencies (CBDC) system called PEReDi. Their contribution is to provide full privacy for anonymous payments (i.e., sender, receiver and transfer value are hidden) and model payment systems using UC framework. PEReDi is based on a threshold blind PS signature [36] and a threshold ElGamal encryption [27]. Their auditability is to revoke sender/receiver's identities and trace anonymous payments. Similarly, our accountability aims to revoke a user and trace multiple showings of that user. We simply put "×" to avoid further explanations in Table 1, but we can put "✓" based on PERdDI's audit protocol. We list three differences here. First, their threshold ElGamal encryption runs twice as their audit protocol aims to identify a user's public key and trace all transactions made by that user. Their threshold blind PS signature doubles signer (or maintainer)'s public-key. Our AC-TIR scheme runs threshold ElGamal encryption once for revocation, and our Σ_{PS} scheme avoids doubling public keys for unforgeability analysis. Second, their tracing algorithm requires the maintainers

to compute the transfer value v from g^v, which requires v to be small. Our AC-TIR scheme does not have such constraint because we do not need to find the undisclosed attribute m from g^m. Third, their construction aims to provide full privacy to payment systems, while our AC-TIR scheme protects user's privacy and reveals the CA's public-key. We do not consider issuer-hiding in this work.

Rathee et al. introduced ZEBRA [35] that allows multiple CAs to issue different credentials that can be efficiently verified on the chain. ZEBRA's anonymity allows anyone to verify the issued and batched credentials from any CA. ZEBRA's revocation is realized via two Merkle trees. Specifically, each CA maintains a Merkle tree for revocation, a master Merkle tree is used to store all CA Merkle trees' roots. For credential verification, a user owns a credential generates two proofs (i.e., a non-membership proof of the user's credential on a CA's Merkle tree, and a membership proof of a CA's root on the master Merkle tree) and a ciphertext on an identifier using a TPKE scheme, where the encryption key is a CA's public key. ZEBRA's accountability requires a threshold number of auditors to decrypt the ciphertext and retrieve the identifier. Then, the auditors inform the specific CA to revoke the specific credential. We notice that ZEBRA leaves the "selective disclose" to potential extensions (Sect 5.4).

Bisht et al. proposed revocable threshold based anonymous credentials over blockchains (RTACO) [6] that follows Coconut′. RTACO is constructed from a dynamic threshold accumulator (DTA) and an anonymous credential scheme DTRAC. We present two points. First, their unblind process in the underlying DTRAC scheme (Appendix A.2) is incorrect. The elements $\widehat{S}$ and $\widehat{Y}$ in equation (15) cannot calculate as they are from different groups. A trivial fix is to double the public key like Coconut′. Second, their execution time (e.g., credential issuance, show and verify) in RTACO is significantly higher than Coconut. This is because the pairing-based DTA is involved in all the phases, such as generate and verify the accumulator (non) membership proofs. Our revocation incurs no pairings except credential verification.

Doerner et al. [25] introduced an AC with threshold issuance. Their main contribution is to thresholdize the BBS+ signature scheme [4] (we call it tBBS+). This tBBS+ scheme requires a single request and response between a user and a group of credential issuers, and the scheme's public-key size is constant compared to [36, 41]. However, the signature used in the tBBS+ scheme is partially blind, i.e., only the message (or attribute) is hidden. Besides, their tBBS+ scheme requires multiple issuers to execute a MPC protocol (every two issuers exchange two messages) to response to a user's issuance request. Our Σ_{PS} signature is strongly-blind, both message and resulting signature are hidden from issuers, and our AC-TIR scheme requires no MPC during credential issuing.

Comparisons to Accountable/Revocable ACs. We show two technical novelties. First, we build a new PS-type signature Σ_{PS}. Our PS-type signature realizes blindness via a linear encryption scheme, which is different from the commitment-based blind PS signatures. Our PS-type signature achieves "short" public key and "fast" threshold issuance. The commitment-based blind PS signature requires the public key size to be two times linear to the number of attributes to achieve unforgeability. Our PS-type signature avoids doubling the issuer's public-key size. Moreover, our threshold issuance runs faster than non-revocable Coconut′s in a setting with 20+ authorities and a credential including 100 attributes. Our implementation confirms that the commitment-based blind PS signature is unsuitable in a large-scale AC, as the credential issuance runs much slower when the authority number grows. Second, we realize revocation at a low cost. Our revocable scheme incurs less computational overhead than ZEBRA and RTACO. Their revocable schemes are based on (threshold) dynamic accumulators; they require (non-revoked) users to compute complex (e.g., pairing-based) membership proofs during credential proof, and some trusted parties to update the accumulator witnesses periodically. Our proof and verification performance remains roughly the same as the non-revocable Coconut/Coconut' in Figure 2.

Comparisons to Issuer-Hiding ACs. Issuer-hiding multi-authority AC schemes [11, 19, 32, 38] enable a user with a credential to show it comes from a set of issuers accepted by a verifier. Mir et al. [32] proposed two signature primitives based on a modified PS signature. We list few differences. First, IhMA focuses on aggregate signatures to achieve a constant-size credential, independent in the number of issuers and attributes. Second, IhMA restricts the number of issuers to be smaller than the number of attributes when defining policies. Third, IhMA considers the issuers must be honest. Our AC-TIR scheme allows the adversary to control less than t issuers, and the number of issuers and attributes are independent. Recently, Sanders and Traore [38] proposed a new PS-type signature to ensure issuer-hiding, compactness and policy audit. But, their IHAC scheme requires a trust assumption in setup (i.e., a pair (g^x, h^x) is made public) and a generic group model (GGM) in unforgeability analysis. Generally, the state-of-the-art issue-hiding ACs have limitations. They lack user revocation, and they require a verifier to choose a set of issuers to define a policy *pol* beforehand so that a user can prove s/he owns a valid credential on disclosed attributes issued independently by issuers in *pol*.

Comparisons to Distributed Identifiers (DIDs). Maram et al. [30] presented CanDID, a decentralized identity system that allows users to manage their own credentials under the self-created decentralized identifiers (DID). CanDID supports legacy-compatible credential issuance, sybil-resistance, accountability and key recovery. We focus on its accountability, which considers traceability and revocation. Traceability allows CanDID committee to link master credentials with context-based ones, and revocation allows CanDID committee to revoke users' credentials via an updated revocation list. The revocation list stores users' self-created public keys (i.e., pseudonyms), and each pseudonym links to a credential. A limitation of CanDID is that the committee can always link a user's different pseudonyms.

Wang et al. [44] introduced Hades, which follows the design of CanDID and ZEBRA. Hades supports full accountability including auditability, traceability and revocation. Auditability means committee members can reveal the identity information of the registered pseudonyms. Traceability means committee members can trace a user's unlinkable pseudonyms. Revocation means committee members can revoke users' credentials via a revocation list. The revocation list stores users' pseudonyms. Hades cannot achieve untraceability, meaning that CAs can trace multiple pseudonym

registrations to a credential generation. Our AC-TIR scheme guarantees such untraceability due to our Σ_{PS} signature's blindness. Also, Hades allows a threshold number of committee members to decrypt the threshold ciphertext twice to realize audit and trace. Our AC-TIR scheme requires a threshold number of authorities to decrypt the threshold ciphertext once to realize revoke.

Property-Wise Comparisons. We focus on anonymous credentials with threshold issuance, threshold revocation, privacy and security (i.e., accountability and unforgeability). Accountability depends on threshold revocation. We choose relevant schemes for comparison in Table 1 due to certain overlaps in designs or building blocks. Selective Disclose means that a user can disclose the encoded attributes selectively. Threshold indicates threshold cryptography is used for issuing, showing, tracing, revoking and auditing AC. Blind Issue means that a user obtains a credential on an attribute m without revealing m to authorities. DID means that whether the scheme supports DID or not. Security analysis considers UC/Game/Simulation/GGM-based and some informal analysis (we use "×" in Table 1).

8 Conclusion

This paper introduced a secure and fast anonymous credential with threshold issuance and revocation (AC-TIR). Our proposed AC-TIR scheme is the best trade-off considering privacy, security, and efficiency in distributed anonymous credentials. Essentially, a user (or credential holder)'s privacy should remain during the prove protocol when s/he is not revoked via a verifier-local revocation technique. Once a user is revoked by a threshold number of authorities, the user's privacy is lost, but the revoked credential remains unforgeable. On the technical side, we introduced a new PS-type signature primitive to construct our AC-TIR, which is of independent interest. We provided detailed benchmarks on state-of-the-art AC schemes and confirmed our efficiency claims on AC-TIR. Finally, we still have several future works to investigate, such as replacing Shamir's SSS with DPSS protocols, replacing NIZK with zk-SNARK (for short proofs), supporting dynamic TPKE schemes and issuer-hiding property.

References

[1] [n. d.]. National Institute of Standards and Technology. NIST IR 8214C – NIST First Call for Multi-Party Threshold Schemes. https://csrc.nist.gov/pubs/ir/8214/c/ipd

[2] 2024. AC-TIR: Fast Anonymous Credentials with Threshold Issuance and Revocation. https://github.com/ACSAC24/AC-TIR.git.

[3] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. 2013. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering* 3, 2 (2013), 111–128.

[4] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-size dynamic k-TAA. In *SCN*. 111–125.

[5] Foteini Baldimtsi and Anna Lysyanskaya. 2013. Anonymous credentials light. In *CCS*. 1087–1098.

[6] Kanchan Bisht, Neel Yogendra Kansagra, Reisha Ali, Mohammed Sayeed Shaik, Maria Francis, and Kotaro Kataoka. 2024. Revocable TACO: Revocable Threshold based Anonymous Credentials over Blockchains. In *AsiaCCS*. 1378–1393.

[7] Johannes Blömer, Jan Bobolz, Denis Diemert, and Fabian Eidens. 2019. Updatable anonymous credentials and applications to incentive systems. In *CCS*. 1671–1685.

[8] Dmytro Bogatov, Angelo De Caro, Kaoutar Elkhiyaoui, and Björn Tackmann. 2021. Anonymous transactions with revocation and auditing in hyperledger fabric. In *CANS*. 435–459.

[9] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short group signatures. In *CRYPTO*, Vol. 3152. 41–55.

[10] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short signatures from the Weil pairing. In *ASIACRYPT*. 514–532.

[11] Daniel Bosk, Davide Frey, Mathieu Gestin, and Guillaume Piolle. 2022. Hidden Issuer Anonymous Credential. *PETs 2022* (2022), 571–607.

[12] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. 2017. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *S&P*. 901–920.

[13] Jan Camenisch, Manu Drijvers, and Jan Hajny. 2016. 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*. 123–133.

[14] Jan Camenisch and Anna Lysyanskaya. 2002. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Crypto*, Vol. 2442. 61–76.

[15] Ran Canetti. 2001. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*. 136–145.

[16] Ran Canetti and Shafi Goldwasser. 1999. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*. 90–106.

[17] Melissa Chase, Trevor Perrin, and Greg Zaverucha. 2020. The signal private group system and anonymous credentials supporting efficient verifiable encryption. In *CCS*. 1445–1459.

[18] David Chaum. 1985. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In *EUROCRYPT*. 241–244.

[19] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. 2022. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In *PKC*. 409–438.

[20] Elizabeth C Crites and Anna Lysyanskaya. 2019. Delegatable anonymous credentials from mercurial signatures. In *CT-RSA*. 535–555.

[21] Ivan Damgård, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. 2021. Balancing privacy and accountability in blockchain identity management. In *CT-RSA*. 552–576.

[22] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. *PoPETs* 2018, 3 (2018), 164–180.

[23] Cécile Delerablée and David Pointcheval. 2008. Dynamic threshold public-key encryption. In *CRYPTO*. 317–334.

[24] David Derler and Daniel Slamanig. 2019. Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Designs, Codes and Cryptography* 87, 6 (2019), 1373–1413.

[25] Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. 2023. Threshold bbs+ signatures for distributed anonymous credential issuance. In *S&P*. 773–789.

[26] Pierre-Alain Fouque and David Pointcheval. 2001. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT*. 351–368.

[27] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. 1999. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*. 295–310.

[28] Ari Juels, Michael Luby, and Rafail Ostrovsky. 1997. Security of blind digital signatures. In *CRYPTO*. 150–164.

[29] Aggelos Kiayias, Markulf Kohlweiss, and Amirreza Sarencheh. 2022. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. In *CCS*. 1739–1752.

[30] Deepak Maram, Harjasleen Malvai, Fan Zhang, Nerla Jean-Louis, Alexander Frolov, Tyler Kell, Tyrone Lobban, Christine Moy, Ari Juels, and Andrew Miller. 2021. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability. In *S&P*. 1348–1366.

[31] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. 2019. CHURP: dynamic-committee proactive secret sharing. In *CCS*. 2369–2386.

[32] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. 2023. Aggregate Signatures with Versatile Randomization and Issuer-Hiding Multi-Authority Anonymous Credentials. In *CCS*. 30–44.

[33] Torben Pryds Pedersen. 1991. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*. 129–140.

[34] David Pointcheval and Olivier Sanders. 2016. Short randomizable signatures. In *CT-RSA*. 111–126.

[35] Deevashwer Rathee, Guru Vamsi Policharla, Tiancheng Xie, Ryan Cottone, and Dawn Song. 2022. Zebra: Anonymous credentials with practical on-chain verification and applications to kyc in defi. *Cryptology ePrint Archive* (2022).

[36] Alfredo Rial and Ania M Piotrowska. 2022. Security analysis of coconut, an attribute-based credential scheme with threshold issuance. *Cryptology ePrint Archive* (2022).

[37] Alfredo Rial and Ania M Piotrowska. 2023. Compact and Divisible E-Cash with Threshold Issuance. *arXiv preprint arXiv:2303.08221* (2023).

[38] Olivier Sanders and Jacques Traoré. 2024. Compact Issuer-Hiding Authentication, Application to Anonymous Credential. *PETS* (2024).

[39] Dominique Schröder and Dominique Unruh. 2012. Security of blind signatures revisited. In *PKC*. 662–679.

[40] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[41] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. 2019. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS*.
[42] Yen-Chieh Sung and Albert Zhang. 2021. DIT: De-Identified Authenticated Telemetry at Scale.
[43] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. 2021. A^{2}L: Anonymous atomic locks for scalability in payment channel hubs. In *S&P*. 1834–1851.
[44] Ke Wang, Jianbo Gao, Qiao Wang, Jiashuo Zhang, Yue Li, Zhi Guan, and Zhong Chen. 2023. Hades: Practical Decentralized Identity with Full Accountability and Fine-grained Sybil-resistance. In *ACSAC*. 216–228.
[45] Zhiyi Zhang, Michał Król, Alberto Sonnino, Lixia Zhang, and Etienne Rivière. 2021. EL PASSO: efficient and lightweight privacy-preserving single sign on. *PETs* (2021).

A Security Analysis of Σ_{PS}

We assume the underlying NIZK proof involved in the Σ_{PS} satisfies the basic security properties: *correctness*, *zero-knowledge* and *simulation-sound extractability*.

Theorem A.1. *The proposed Σ_{PS} satisfies unlinkability under the indistinguishability (IND) definition, unforgeability under the assumption Ass, and untraceability under the DLIN assumption.*

Proof. *For unforgeability.* Let $\mathcal{S}$ denote an assumption *Ass* solver, who is given (h, h^x, h^y) and an oracle $O(m)$, aims to compute (g, g^{x+m^*y}). $\mathcal{S}$ sets up an unforgeablity game for $\mathcal{A}$ by creating a signer with public key $\mathsf{pk} = (h, h^x, h^y)$. When $\mathcal{A}$ issues a signing query regards a message m and a user's public key $\mathsf{pk}_U = (u, v, w)$, $\mathcal{S}$ forwards the message m to his oracle $O(m)$ and receives (g, g^{x+my}), where $w = u^{d_1} = v^{d_2}$ and $\hat{e}(g, h^x \cdot h^{ym}) = \hat{e}(g^{x+my}, h)$. Then, $\mathcal{S}$ simulates the blinded PS signature as $\tilde{U} = u^{r'_1}, \tilde{V} = v^{r'_2}, \tilde{W} = g^{x+my} \cdot w^{r'_1+r'_2}$, where $(r'_1, r'_2) \in \mathbb{Z}_p^2$. *This step is the main reason of using linear encryption scheme. $\mathcal{A}$ obtains a valid PS signature (g, g^{x+my}) using the secret key $\mathsf{sk}_U = (d_1, d_2)$ (see Figure 1). Coconut cannot simulate its blind PS signature and Coconut′ simulates it by extending public key (i.e., add values such as g^y).* At some point, $\mathcal{A}$ outputs a forgery (m^*, g, g^{x+m^*y}). $\mathcal{S}$ checks where the forgery is valid under pk and the forgery was not previously simulated by $\mathcal{S}$. If all checks hold, $\mathcal{S}$ uses this forgery to break assumption *Ass*.

For untraceability/blindness. Let $\mathcal{S}$ denote a DLIN assumption solver, who is given $(u, v, w, u^a, v^b, w^{a+b})$, aims to decide w^{a+b} from a random value w^z, where $z \in \mathbb{Z}_p$. $\mathcal{S}$ relies on the given DLIN instances to simulate the blindness game for $\mathcal{A}$ as follows [28]. $\mathcal{S}$ creates two users (e.g., *left* user and *right* user) and a signer with a key pair $(\mathsf{sk}, \mathsf{pk})$, where $\mathsf{sk} = (x, y)$. $\mathcal{S}$ sends sk to $\mathcal{A}$ and chooses $\tilde{b} = [0, 1]$. When $\mathcal{A}$ initiates two user interactions with two messages (m_0^*, m_1^*), $\mathcal{S}$ simulates an interactive signing algorithm regarding the *left* user (i.e., $User(\mathsf{pk}, m_{\tilde{b}})$) as follows. $\mathcal{S}$ first sets the user's public key as $\mathsf{pk}_U = (u, v, w)$, and creates a group element $g \in \mathbb{G}$, and simulates a ciphertext $(U, V, W) = (u^a, v^b, w^{a+b} \cdot g^{m_{\tilde{b}}})$. Since the values (a, b) are unknown to $\mathcal{S}$ when generating a NIZK proof π_s, $\mathcal{S}$ simulates the NIZK proofs on (a, b) using the approach described in the DL-based sigma protocol. Eventually, $\mathcal{S}$ returns $(\mathsf{pk}_U, g, U, V, W, \pi_s)$ to $\mathcal{A}$.

Upon receiving $(\mathsf{pk}_U, g, U, V, W, \pi_s)$ from $\mathcal{S}$, $\mathcal{A}$ verifies π_s and creates a blinded signature $(\tilde{U}, \tilde{V}, \tilde{W})$ using sk. Eventually, $\mathcal{S}$ outputs an unblinded PS signature regards the *left* user as $(g, g^{x+m_{\tilde{b}}y})$. Now, we analyze $\mathcal{A}$'s probability of deciding the ciphertext (U, V, W). We assume $\mathcal{A}$ chooses a bit b', and checks the following equation

$$\hat{e}(w^{a+b} \cdot g^{m_{\tilde{b}}} / g^{m_{\tilde{b}'}}, h) = \hat{e}(w^{a+b}, h) \cdot \hat{e}(g^{m_{\tilde{b}} - m_{\tilde{b}'}}, h).$$

It is easy to see that the pairing holds in case of $\tilde{b} = \tilde{b}'$ and $\tilde{b} \neq \tilde{b}'$. Similarly, $\mathcal{S}$ simulates another interactive signing algorithm $User(\mathsf{pk}, m_{1-\tilde{b}})$ regards the *right* user using the same method except replacing w^{a+b} in W with w^z. The above equation also holds. If $\mathcal{A}$ guesses the random bit correctly, i.e., $\tilde{b}' = \tilde{b}$, then $\mathcal{S}$ can solve the DLIN problem.

For unlinkability. The reduction refers to Lemma B.6.

□

B Security Analysis of AC-TIR

B.1 Privacy

Theorem B.1. *The proposed scheme is privacy-preserving if the NIZK is zero-knowledge, the commitment scheme is hiding, the threshold Elgamal encryption scheme is CPA-secure, DLIN assumption is held in the asymmetric pairing, and the IND definition is sound.*

Proof. We define a sequence of games $\mathbb{G}_i$, $i = [0, 7]$ and let $\mathcal{A}$ denote the advantage of the adversary in game $\mathbb{G}_i$. We assume that $\mathcal{A}$ issues at most q_0 issue queries and q_1 prove queries at each game. We assume each issue query involves a single attribute for simplicity.

- $\mathbb{G}_0$: This is the original privacy game.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_0$ except the following difference: $\mathcal{S}$ randomly chooses $g_0 \in [1, q_0]$ as a guess for the index of the challenge query with respect to two attributes (m_0^*, m_1^*) during the issue protocol. Similarly, $\mathcal{S}$ chooses $g_1 \in [1, q_1]$ as a guess for the index of the challenge query with respect to two attributes (m_0^*, m_1^*) during the prove protocol. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s challenge query does not occur in this g_0/g_1-th query. Since q_0/q_1 queries are involved in the game, we have
$$\mathsf{Adv}_0 = q_0 \cdot q_1 \cdot \mathsf{Adv}_1 \tag{1}$$
- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except that $\mathcal{S}$ outputs a random bit if a $\mathbf{Guess_1}$ event occurs in the g_0-th query. Then we have:
$$|\mathsf{Adv}_1 - \mathsf{Adv}_2| \leq \Pr[\mathbf{Guess}_1] \tag{2}$$

Lemma B.2. *The $\mathbf{Guess}_1$ event happens only with a negligible probability when the commitment scheme achieves hiding property.*

Let $\mathcal{S}$ denote a simulator, who is choosing a random bit $b \in [0, 1]$, aims to play a privacy game with $\mathcal{A}$. $\mathcal{S}$ simulates the privacy game for $\mathcal{A}$ as follows.

- $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating n authorities along with a key pair $(\mathsf{sk}, \mathsf{pk})$, where $\mathsf{sk} = (x, y)$, $\mathsf{pk} = (h^x, h^y)$. Each authority owns a key share $(\mathsf{sk}_i, \mathsf{pk}_i)$, where $\mathsf{sk}_i = (x_i, y_i)$, $\mathsf{pk} = (h^{x_i}, h^{y_i})$. Also, $\mathcal{S}$ sets up the commitment scheme's public parameter as par. $\mathcal{S}$ returns (pk, par) to $\mathcal{A}$, and publishes an initially empty set VLR.
- When $\mathcal{A}$ issues an issue query regards a user, an attribute m, and a statement ϕ such that $\phi(m) = 1$, $\mathcal{S}$ simulates a credential σ for $\mathcal{A}$. Specifically, the credential σ is a Σ_{PS} signature on the hashed message $c_m \leftarrow \mathsf{Com}(par, m)$. $\mathcal{S}$ can answer prove and revoke queries honestly according to the protocol specification.
- If $\mathcal{A}$ submits two challenge attributes $(m_0^*, m_1^*) \notin VLR$ and a statement ϕ^* such that $\phi^*(m_b^*) = 1$, $\mathcal{S}$ simulates a commitment c_m^* based on b, where $(c_m^*, open) \leftarrow \mathsf{Com}(par, m_b^*)$. Note that

either m_0^* or m_1^* can be previously-queried, and *open* refers to the randomness used in generating c_m^*.
If $\mathcal{A}$ guesses the random bit correctly, then $\mathcal{A}$ breaks the commitment scheme's hiding property. Therefore, we have

$$\Pr[\mathbf{Guess}_1] \leq \mathsf{Adv}_{\mathcal{A}}^{\mathrm{Hide}}(\lambda) \tag{3}$$

- $\mathbb{G}_3$: This game is identical to game $\mathbb{G}_2$ except that $\mathcal{S}$ outputs a random bit if a $\mathbf{Guess}_2$ event occurs in the g_0-th query. Then we have:

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq \Pr[\mathbf{Guess}_2] \tag{4}$$

Lemma B.3. *The* $\mathbf{Guess}_2$ *event happens only with a negligible probability if DLIN assumption holds in the asymmetric pairing.*

Let $\mathcal{S}$ denote a DLIN assumption solver, who is given $(u, v, w, u^a, v^b, w^{a+b})$, aims to decide w^{a+b} from a random value. $\mathcal{S}$ relies on the given DLIN instances to simulate the privacy game for $\mathcal{A}$ as follows.
- $\mathcal{S}$ performs the same setup as described in game $\mathbb{G}_2$. $\mathcal{S}$ chooses $\tilde{b} = [0, 1]$, which will be used in the following game. In this game, we give $\mathcal{A}$ a subset of key shares $\{\mathsf{sk}_i\}$ in order to simulate the interactive issue protocol between $\mathcal{A}$ and $\mathcal{S}$, where $i \in [1, t)$.
- When $\mathcal{A}$ issues an issue query regards two attributes $(m_0^*, m_1^*) \notin VLR$, and a statement ϕ^* such that $\phi^*(m_{\tilde{b}}^*) = 1$, $\mathcal{S}$ simulates an interactive issue protocol between the first user and i-th authority, i.e., $User(\mathsf{pk}_i, m_{\tilde{b}}^*)$ as follows. $\mathcal{S}$ first sets the user's key pair as $(\mathsf{sk}_U, \mathsf{pk}_U)$, creates a commitment c_m from m_0^* and a generator $\tilde{g} = \mathsf{H}_1(c_m)$, and simulates a ciphertext $(U, V, W) = (u^a, v^b, w^{a+b} \cdot \tilde{g}^{m_{\tilde{b}}})$. Since the values (a, b) are unknown to $\mathcal{S}$ when generating π_s, $\mathcal{S}$ simulates the NIZK proofs on (a, b) using the approach described in the DL-based sigma protocol. Eventually, $\mathcal{S}$ returns a tuple $(\mathsf{pk}_U, c_m, \tilde{g}, U, V, W, \pi_s, \phi)$ to $\mathcal{A}$. Upon receiving the tuple from $\mathcal{S}$, $\mathcal{A}$ verifies $\tilde{g}$ using c_m and π_s, and signs (U, V, W) using $\mathsf{sk}_i = (x_i, y_i)$. Afterwards, $\mathcal{S}$ simulates a Σ_{PS} signature $(\tilde{g}, \tilde{U}, \tilde{V}, \tilde{W})$ using the same method described in Theorem A.1 (for unforgeability), and outputs the Σ_{PS} signature $(\tilde{g}, \tilde{g}^{x_i + m_{\tilde{b}} y_i})$ regards the first user. Now, we analyze $\mathcal{A}$'s probability of deciding the ciphertext $(u^a, v^a, w^{a+b} \cdot \tilde{g}^{m_{\tilde{b}}})$, we assume $\mathcal{A}$ chooses a bit $\tilde{b}'$, and checks the following equation

$$\hat{e}(w^{a+b} \cdot \tilde{g}^{m_{\tilde{b}}} / \tilde{g}^{m_{\tilde{b}'}}, h) = \hat{e}(\underline{w^{a+b}}, h) \cdot \hat{e}(\tilde{g}^{m_{\tilde{b}} - m_{\tilde{b}'}}, h).$$

It is easy to see that the pairing holds in case of $\tilde{b} = \tilde{b}'$ and $\tilde{b} \neq \tilde{b}'$. Similarly, $\mathcal{S}$ simulates another interactive issue protocol between $User(\mathsf{pk}_i, m_{1-\tilde{b}})$ and i-th authority using the same method described above. Eventually, if $\mathcal{A}$ guesses the random bit correctly, i.e., $\tilde{b}' = \tilde{b}$, then $\mathcal{S}$ can solve the DLIN problem. Therefore, we have

$$\Pr[\mathbf{Guess}_2] \leq \mathsf{Adv}_{\mathcal{S}}^{\mathrm{DLIN}}(\lambda) \tag{5}$$

- $\mathbb{G}_4$: This game is identical to game $\mathbb{G}_3$ except that $\mathcal{S}$ outputs a random bit if a $\mathbf{Guess}_3$ event occurs in the g_0-th query. Then we have:

$$|\mathsf{Adv}_3 - \mathsf{Adv}_4| \leq \Pr[\mathbf{Guess}_3] \tag{6}$$

Lemma B.4. *The* $\mathbf{Guess}_3$ *event happens only with a negligible probability when NIZK is zero-knowledge.*

Let $\mathcal{S}$ denote an attacker, who is given a common random string crs and two proof generation oracles P_b, aims to break the NIZK's zero-knowledge (ZK) property. Note that $b \in [0, 1]$ is unknown to $\mathcal{S}$. $\mathcal{S}$ simulates the privacy game for $\mathcal{A}$ as follows.
- $\mathcal{S}$ performs the same setup as described in game $\mathbb{G}_2$.
- When $\mathcal{A}$ submits two messages (m_0^*, m_1^*) and a statement ϕ^* that satisfies $\phi^*(m_b^*) = 1$. Then, $\mathcal{S}$ performs the following steps. First, $\mathcal{S}$ forwards (ϕ^*, m_0^*) to its proof generation oracle P_0 and receives a proof π_0^* associated with the witness m_0^*. $\mathcal{S}$ simulates another proof π_1^* associated with the witness m_1^* using the same method. Second, $\mathcal{S}$ interacts with $\mathcal{A}$ to derive two Σ_{PS} signatures $(s_{i,0}, s_{i,1})$ using (sk_i, m_0^*) and (sk_i, m_1^*), respectively, where index i indicates the partial Σ_{PS} signature from i-th authority. Eventually, $\mathcal{S}$ returns one of signatures $(\sigma_0, \sigma_1) = [(s_0, \pi_0^*), (s_1, \pi_1^*)]$ to $\mathcal{A}$, where (s_0, s_1) indicates the aggregated Σ_{PS} signatures.
$\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. If $\mathcal{A}$ guesses the random bit correctly, $\mathcal{S}$ can break the NIZK's zero-knowledge property. Therefore, we have

$$\Pr[\mathbf{Guess}_3] \leq \mathsf{Adv}_{\mathcal{S}}^{\mathrm{ZK}}(\lambda) \tag{7}$$

- $\mathbb{G}_5$: This game is identical to game $\mathbb{G}_4$ except that $\mathcal{S}$ outputs a random bit if a $\mathbf{Guess}_4$ event occurs in the g_1-th query. Then we have:

$$|\mathsf{Adv}_4 - \mathsf{Adv}_5| \leq \Pr[\mathbf{Guess}_4] \tag{8}$$

Lemma B.5. *The* $\mathbf{Guess}_4$ *event happens only with a negligible probability if the threshold ElGamal scheme is CPA-secure.*

Let $\mathcal{S}$ denote an attacker, who is given a public key pk, a subset of key shares $\{i, \mathsf{sk}_i\}$ where $i \in [1, t)$, and a challenge oracle, aims to break the threshold ElGamal's CPA property. $\mathcal{S}$ simulates the privacy game for $\mathcal{A}$ as follows.
- $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating n authorities, and sets the authorities' common public key as pk. Also, $\mathcal{S}$ sets up the commitment scheme's public parameter as par. $\mathcal{S}$ returns (pk, par) to $\mathcal{A}$, and creates an initially empty set VLR. In this game, we give $\mathcal{A}$ a subset of key shares $\{i, \mathsf{sk}_i\}$ where $i \in [1, t)$ in order to answer less than t corrupt queries toward authorities.
- During training stage, if $\mathcal{A}$ issues an issue query regards an attribute m to $\mathcal{S}$, $\mathcal{S}$ simulates a valid Σ_{PS} signatures using the method described in the unforgeability game of PS signatures [34] because $\mathcal{S}$ does not know the secret key sk. $\mathcal{S}$ can answer $\mathcal{A}$'s revoke queries easily.
- At the challenge stage, when $\mathcal{A}$ issues a prove query regards two attributes (m_0^*, m_1^*) and a statement ϕ^* satisfying $\phi^*(m_b^*) = 1$, $\mathcal{S}$ forwards (m_0^*, m_1^*) to his challenger and receives a ciphertext $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^*)$. $\mathcal{S}$ simulates the NIZK proofs on the secret randomness k_0 using the approach described in the DL-based sigma protocol, and returns the proof $\Theta = (\cdots, C^*)$ to $\mathcal{A}$.
If $\mathcal{A}$ guesses the random bit correctly, i.e., $b' = b$, then $\mathcal{S}$ can break the threshold ElGamal's CPA security. Therefore, we have

$$\Pr[\mathbf{Guess}_4] \leq \mathsf{Adv}_{\mathcal{S}}^{\mathrm{CPA}}(\lambda) \tag{9}$$

- $\mathbb{G}_6$: This game is identical to game $\mathbb{G}_5$ except that $\mathcal{S}$ outputs a random bit if a $\mathbf{Guess}_5$ event occurs in the g_1-th query. The

simulation is exactly same as game $\mathbb{G}_4$, we have

$$|\mathsf{Adv}_5 - \mathsf{Adv}_6| \leq \mathsf{Adv}^{\mathrm{ZK}}_{\mathcal{S}}(\lambda) \tag{10}$$

- $\mathbb{G}_7$: This game is identical to game $\mathbb{G}_6$ except that $\mathcal{S}$ outputs a random bit if a $\mathbf{Guess}_6$ event occurs in the g_1-th query. Then we have:

$$|\mathsf{Adv}_6 - \mathsf{Adv}_7| \leq \Pr[\mathbf{Guess}_6] \tag{11}$$

Lemma B.6. *The* $\mathbf{Guess}_6$ *event happens only with a negligible probability when IND definition is correct.*

Let $\mathcal{S}$ denote a simulator, who is choosing a random bit $b \in [0,1]$, aims to play a privacy game with $\mathcal{A}$. $\mathcal{S}$ simulates the privacy game for $\mathcal{A}$ as follows.

– $\mathcal{S}$ performs the same setup as described in game $\mathbb{G}_2$. During training stage, $\mathcal{S}$ can simulate issue, prove, corrupt and revoke queries easily.
– At the challenge stage, if $\mathcal{A}$ submits two attributes $(m_0^*, m_1)^* \notin VLR$ and a statement ϕ^* such that $\phi^*(m_b^*) = 1$. Then, $\mathcal{S}$ simulates a Σ_{PS} signature $(\tilde{g}', s', \kappa)$ and creates a proof π_v^* based on m_b^*, where $\tilde{g}' = \tilde{g}^{r'}, s' = \tilde{g}^{(r+x+y\cdot m_b^*)\cdot r'}, \kappa = \alpha \cdot \beta^{m_b^*} \cdot h^r, \pi_v^*$. Note that b is known to $\mathcal{S}$ only. To analyze $\mathcal{A}$'s winning probability, we assume $\mathcal{A}$ chooses a bit b', and checks the following equation

$$\hat{e}(s'/\underline{\tilde{g}^{r'\cdot y_i \cdot m_{b'}^*}}, h) = \hat{e}(\tilde{g}', \kappa/\underline{\beta_i^{m_{b'}^*}}).$$

Since we consider $\mathcal{A}$ as a subset of authorities, $\mathcal{A}$ computes the underlined parts using $\{y_i, \beta_i = h^{y_i}\}$ where $i \in [1, t)$, and checks the above equation. It is easy to see that the pairing holds in case of $b = b'$ and $b \neq b'$.
If $\mathcal{A}$ guesses the random bit correctly, i.e., $b' = b$, then $\mathcal{S}$ breaks the IND definition. Therefore, we have

$$\Pr[\mathbf{Guess}_6] \leq \mathsf{Adv}^{\mathrm{IND}}_{\mathcal{A}}(\lambda) \tag{12}$$

Combining the above results together, we have

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}(\lambda) \quad \leq \quad & q_0 \cdot q_1 \cdot [\mathsf{Adv}^{\mathrm{Hide}}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathrm{DLIN}}_{\mathcal{S}}(\lambda) \\ & + \mathsf{Adv}^{\mathrm{CPA}}_{\mathcal{S}}(\lambda) + \mathsf{Adv}^{\mathrm{IND}}_{\mathcal{A}}(\lambda) + 2\mathsf{Adv}^{\mathrm{ZK}}_{\mathcal{S}}(\lambda)].\end{aligned}$$

□

B.2 Security

Theorem B.7. *The proposed scheme is secure if the NIZK is simulation-sound extractable, the commitment scheme is binding, and the assumption Ass is held in the asymmetric pairing.*

Proof. We define a sequence of games $\mathbb{G}_i$, $i = [0,5]$ and let $\mathcal{A}$ denote the advantage of the adversary in game $\mathbb{G}_i$. We assume that $\mathcal{A}$ issues at most q_0 issue queries and q_1 prove queries at each game. We assume each issue query involves a single attribute for simplicity.

- $\mathbb{G}_0$: This is the original security game.
- $\mathbb{G}_1$: This game is identical to game $\mathbb{G}_0$ except the following difference: $\mathcal{S}$ randomly chooses $g_1 \in [1, q_1]$ as a guess for the index of the forgery which returns a proof $(\Theta^*, \phi^*, VLR^*)$ during the prove protocol, where $\Theta^* = (\kappa^*, \tilde{g}^{*'}, s^{*'}, \pi_v^*)$. Similarly, $\mathcal{S}$ chooses $g_0 \in [1, q_0]$ as a guess for the index of the forgery which returns $(\Lambda^*, \phi^*) = (\pi_s^*, \cdots)$ during the issue protocol. $\mathcal{S}$ will output a random bit if $\mathcal{A}$'s forgery does not occur in this g_0/g_1-th query. Therefore, we have

$$\mathsf{Adv}_0 = q_0 \cdot q_1 \cdot \mathsf{Adv}_1 \tag{13}$$

- $\mathbb{G}_2$: This game is identical to game $\mathbb{G}_1$ except that $\mathcal{S}$ will output a random bit if $\mathbf{Forge}_1$ event happens in the g_1-th query where $\mathcal{A}$ outputs a forgery in the form of $(\Theta^*, \phi^*, VLR^*)$, such that the NIZK proof π_v^* in Θ^* is not returned by the SIM oracle. Then we have:

$$|\mathsf{Adv}_1 - \mathsf{Adv}_2| \leq \Pr[\mathbf{Forge}_1] \tag{14}$$

Lemma B.8. *The* $\mathbf{Forge}_1$ *event happens only with a negligible probability when the NIZK is simulation-sound extractable.*

Let $\mathcal{S}$ denote an attacker against simulation-sound extractability property, who is given a common random string crs and a simulation oracle SIM, aims to output (ϕ^*, π_v^*), i.e., a proof π_v^* for a statement ϕ^* associated with a witness m^*. $\mathcal{S}$ simulates the security game for $\mathcal{A}$ as follows:

– $\mathcal{S}$ sets up the game for $\mathcal{A}$ by creating ℓ users, and n authorities along with a key pair $(\mathsf{sk}, \mathsf{pk})$, where $\mathsf{sk} = (x, y)$, $\mathsf{pk} = (h^x, h^y)$. Each authority owns a key share $(\mathsf{sk}_i, \mathsf{pk}_i)$, where $\mathsf{sk}_i = (x_i, y_i)$, $\mathsf{pk} = (h^{x_i}, h^{y_i})$. Also, $\mathcal{S}$ sets up the commitment scheme's public parameter as par. $\mathcal{S}$ returns $(\mathsf{pk}, \mathrm{crs})$ to $\mathcal{A}$, and creates an initially empty set VLR.
– When $\mathcal{A}$ issues a prove query regards a statement ϕ and a message m satisfying $\phi(m) = 1$, $\mathcal{S}$ forwards ϕ to his simulation oracle SIM and receives a proof π associated with the witness m. Then, $\mathcal{S}$ simulates a valid Σ_{PS} signature using sk and m, and returns $\Theta = (\kappa, \tilde{g}', s', \pi)$ to $\mathcal{A}$. $\mathcal{S}$ answers the issue queries using the similar method except that $\mathcal{S}$ relies on authorities' key shares to derive partial Σ_{PS} signatures. Also, $\mathcal{S}$ answers corrupt and revoke queries easily.
– At some point, $\mathcal{A}$ outputs a forgery $(\Theta^*, \pi_v^*) = (\kappa^*, \tilde{g}^{*'}, s^{*'}, \pi_v^*)$ and a revocation list VLR^*. $\mathcal{S}$ checks: 1) the Σ_{PS} signature $(\kappa^*, \tilde{g}^{*'}, s^{*'})$ is verified as valid under (pk, VLR^*); 2) the proof π_v^* for statement ϕ^* is verified as valid; 3) (ϕ^*, π_v^*) was not returned by SIM oracle. If all these conditions hold, $\mathcal{S}$ uses (ϕ^*, π_v^*) to break the simulation-sound extractability property. Therefore, we have

$$\Pr[\mathbf{Forge}_1] \leq \mathsf{Adv}^{\mathrm{Ext}}_{\mathcal{S}}(\lambda) \tag{15}$$

- $\mathbb{G}_3$: This game is identical to game $\mathbb{G}_2$ except that $\mathcal{S}$ will output a random bit if $\mathbf{Forge}_2$ event happens in the g_0-th query where $\mathcal{A}$ outputs a forgery in the form of $\Lambda^* = (\pi_s^*, \cdots)$, such that the NIZK proof π_s^* in Λ^* is not returned by the SIM oracle. By using the similar reduction described in game $\mathbb{G}_2$, we have:

$$|\mathsf{Adv}_2 - \mathsf{Adv}_3| \leq \mathsf{Adv}^{\mathrm{Ext}}_{\mathcal{S}}(\lambda) \tag{16}$$

- $\mathbb{G}_4$: This game is identical to game $\mathbb{G}_3$ except that $\mathcal{S}$ will output a random bit if a collision event happens in the g_0-th query where $\mathcal{A}$ outputs $(\Lambda^*, \Lambda^{*'})$ satisfying $c_m = c'_m$ during the issue protocol. However, the extracted witnesses from $(\pi_s^*, \pi_s^{*'})$ satisfy $m^* \neq m^{*'}$. By using the binding property of the commitment scheme, we have:

$$|\mathsf{Adv}_3 - \mathsf{Adv}_4| \leq \mathsf{Adv}^{\mathrm{Bind}}_{\mathcal{S}}(\lambda) \tag{17}$$

- $\mathbb{G}_5$: This game is identical to game $\mathbb{G}_4$ except that $\mathcal{S}$ will output a random bit if $\mathbf{Forge}_3$ event happens in the g_0-th query where

$\mathcal{A}$ outputs a forgery in the form of $(\Theta^*, \phi^*, VLR^*)$, such that the Σ_{PS} signature $(\kappa^*, \tilde{g}^{*'}, s^{*'})$ is not simulated by $\mathcal{S}$. Then we have:

$$|\mathsf{Adv}_1 - \mathsf{Adv}_2| \leq \Pr[\mathbf{Forge}_3] \tag{18}$$

Lemma B.9. *The* $\mathbf{Forge}_3$ *event happens only with a negligible probability when the assumption Ass is held.*

Let $\mathcal{S}$ denote an assumption *Ass* solver, who is given (h, h^x, h^y) and an oracle $O(m)$, aims to compute (g, g^{x+m^*y}). $\mathcal{S}$ simulates the security game for $\mathcal{A}$ as follows:

- $\mathcal{S}$ performs the same setup as described in $\mathbb{G}_2$. Also, $\mathcal{S}$ creates a trapdoor information τ for NIZK system.
- During the training stage, when $\mathcal{A}$ queries $c_m = g^o \cdot g_0^m$ to the random oracle H_1, $\mathcal{S}$ picks $z \in \mathbb{Z}_p$ and sets the hash output as $\tilde{g} = \mathsf{H}_1(c_m) = g^z$. We assume a single attribute m regarding a generator g_0. When $\mathcal{A}$ issues an issue query regards a statement ϕ and a message m, $\mathcal{S}$ forwards the message m to his oracle $O(m)$ and receives (g, g^{x+my}). First, $\mathcal{S}$ computes c_m, and obtains the hash output $\tilde{g} = g^z$ from random oracle H_1. Second, $\mathcal{S}$ simulates the blinded Σ_{PS} signature $(\tilde{U}, \tilde{V}, \tilde{W})$ using the same method described in Theorem A.1 (for unforgeability). Third, $\mathcal{S}$ returns a valid Σ_{PS} signature $(\tilde{g}, s)$ to $\mathcal{A}$, where $s = g^{z(x+my)}$.
  When $\mathcal{A}$ issues a prove query regards a statement ϕ and a message m, $\mathcal{S}$ forwards the message m to his oracle $O(m)$ and receives (g, g^{x+my}). $\mathcal{S}$ first simulates π_v using $(\mathsf{crs}, \tau, \phi)$ when $\phi(m) = 1$. Second, $\mathcal{S}$ computes c_m, and obtains the hash output $\tilde{g} = g^z$ from random oracle H_1. Third, $\mathcal{S}$ chooses two randomness $(r, r') \in \mathbb{Z}_p^2$, and simulates $\tilde{g}' = \tilde{g}^{r'}$, $s' = (\tilde{g}^{x+my} \cdot \tilde{g}^r)^{r'}$, and $\kappa = \alpha \cdot \beta^m \cdot h^r$. Eventually, $\mathcal{S}$ returns $(\tilde{g}', s', \kappa, \pi_v)$ to $\mathcal{A}$.
- At some point, $\mathcal{A}$ outputs a forgery $(\Theta^*, \phi^*, VLR^*)$ and a NIZK proof π_v^*, where $\tilde{g}^{*'} = \tilde{g}^{*r'^*}$, $s^{*'} = (\tilde{g}^{*(x+m^*y)} \cdot \tilde{g}^{*r^*})^{r'^*}$) and $\kappa^* = \alpha \cdot \beta^{m^*} \cdot h^{r^*}$. $\mathcal{S}$ extracts the randomness r^* from π_v^* due to its simulation-sound extractability. Then, $\mathcal{S}$ checks the following
  * The $\mathbf{Forge}_3$ event happens at g_0-th query;
  * The Σ_{PS} signature $(\tilde{g}^{*'}, s^{*'}, \kappa^*)$ and the NIZK proof π^* for statement ϕ^* are verified as valid under (pk, VLR^*);
  * The forgery was not previously simulated by $\mathcal{S}$.

  If all the above conditions hold, $\mathcal{S}$ confirms that it is a successful forgery, $\mathcal{S}$ computes the solution $(g^{r'^*}, g^{(x+m^*y)r'^*})$. Therefore, we have

$$\Pr[\mathbf{Forge}_3] \leq \mathsf{Adv}_{\mathcal{S}}^{\mathrm{Ass}}(\lambda) \tag{19}$$

Combining the above results together, we have

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) \quad \leq \quad q_0 \cdot q_1 \cdot [2\mathsf{Adv}_{\mathcal{S}}^{\mathrm{Ext}}(\lambda) + \mathsf{Adv}_{\mathcal{S}}^{\mathrm{Bind}}(\lambda) + \mathsf{Adv}_{\mathcal{S}}^{\mathrm{Ass}}(\lambda)].$$

□

C Real Applications

Our proposed AC-TIR is suitable for anonymous attestation. The application scenario simply consists of a user (or host) equipped with a tamper-resistant TPM, multiple authorities and a verifier. We assume multiple authorities share the same public key, and each authority owns a key share of the private key. During a credential issuance, a user obtains a credential encoding certain system components (e.g., OS type, version, software state, etc) from a threshold number of independent authorities. We consider user's revocation tag m_0 as TPM attestation key. During a credential showing, the TPM and the user create a valid proof using m_0 and the issued credential for attestation (e.g., attesting the software state). For privacy, multiple showings will not reveal the identity of the user and the TPM. For security, multiple showings from the revoked user (due to exposed attestation keys or credentials) should be rejected by the verifier. Technically, our AC-TIR scheme is close to DAA with revocation capabilities [12] (i.e., the enhanced privacy ID EPID).

Our proposed AC-TIR is user friendly. In the context of attestation, the TPM creates Schnorr's zero-knowledge proofs using m_0, and the user performs the remaining computations for issuance and showings respectively. For example, a show protocol takes 180ms to prove (undisclosed) 100 attributes according to our implementation. There are two revocation techniques used in the AC systems. One approach is (dynamic) accumulator. Roughly, the user must prove in zero-knowledge that their credentials are not contained in a revocation list, such as ZEBRA [35]. Some authorities must update the accumulator witnesses periodically. Another is called verifier local revocation. The computational cost of the user is independent of the number of revoked users. But, the verifier's computational complexity is linear in this number. Thus, our proposed scheme is suitable for the resource-constraint users, and the verifier can be a powerful third party. We notice that Camenish et al. [13] introduced a revocable AC scheme based on Boneh-Boyen signature [9], and the verifier's complexity is logarithmic in the number of revoked users. We leave the construction of AC scheme whose verification time is not linear to the number of revoke users as our future work.

Our proposed AC-TIR is blockchain friendly. The authorities can form a committee in an open blockchain. Later, a threshold number of authorities rely on PoW/PoS consensus to record/revoke credentials during the issue/show protocol. The committee can be dynamic and the majority of committee members must be honest. Dynamic implies that some authorities may leave or join in, and the left authorities can be treated as corrupted parties. We can use DPSS protocol to guarantee such security [31] for our future work. Third, the revocation list can be dynamic in the sense that the authorities can periodically delete the expired revocation tags, which avoids the ever-growing revocation list. The time interval (or epoch)-based revocation can be used in the attestation schemes [13].