Always by Your Side: Constructing Traceable Anonymous Credentials with Hardware-Binding

Chang Chen
Beijing Jiaotong University

Guoyu Yang and Qi Chen
Guangzhou University

Wei Wang
Beijing Jiaotong University

Jin Li
Guangzhou University

***Abstract*—With the development of decentralized identity (DID), anonymous credential (AC) technology, as well as its traceability, is receiving more and more attention. Most works introduce a trusted party (regulator) that holds a decryption key or backdoor to directly deanonymize the user identity of anonymous authentication. While some cryptographic primitives can help regulators handle complex tracing tasks among large amounts of user profiles (stored by the issuer) and authentication records (stored by the service provider), additional security primitives are still needed to ensure the privacy of other users. Besides, hardware-binding anonymous credential (hbAC) systems have been proposed to prevent credential sharing or address platform resource constraints, the traceability of hbAC has yet to be discussed.**

In this paper, we introduce a public key encryption with equality test as a regulatory text for each authentication record to address the above-mentioned challenges. The security of this feature is guaranteed by the verifiability, non-frameability, and round isolation of the proposed scheme. We compared the asymptotic complexity of our scheme with other traceable AC schemes and shows our scheme has advantages in tracing tasks as well as securely outsourcing them. The key feature of our scheme is that the ability of equality test of regulatory texts is independent of the public key, but rather depends on the round identifier of the authentication. We instantiate a traceable, hardware-binding AC scheme based on smart cards and BBS+ signature and give the performance analysis of it.

1. Introduction

The anonymous credential system is a powerful cryptographic tool that enables a *user* to prove possession of certain attributes authorized by a legitimate *issuer* to a service provider (*verifier*) in an anonymous and unlinkable manner. Its privacy and functionality promise applications in numerous authentication-related tasks, such as anonymous voting, auctions, and wireless communications [1], [2], etc.

However, in real-world scenarios, many tasks demand stricter requirements on the identity of entities; for example, in a presidential election, in-person voting is mandatory. As a digital certificate, anonymous credentials cannot guarantee this feature. A feasible approach is separating the credentials between the user and specific hardware, (such as the core/helper setting in [3], and the card/holder setting in [4]), which we called hardware-binding anonymous credentials (hbAC). We follow their settings that the hardware we mentioned always refers to secure hardware whose internal data cannot be read or maliciously tampered with. The main difference is that the helper is responsible for all communications with the service provider under the former setting, while the latter allows the hardware (card) to communicate with the service provider. These approaches help mitigate application risks on the user side, like preventing credential redistribution, while also offloading computationally intensive tasks onto standard devices with more robust computational and storage resources, such as smartphones. During credential usage, the helper is responsible for communicating externally and calculating the primary content of the credential based on usage requirements. Additionally, the helper interacts with the core, which stores the other part of the credential's critical information, to compute the complete proof of credential possession jointly.

A typical application of hbAC is Direct Anonymous Attestation (DAA) [5], which is a widely used scheme on Windows. DAA provides remote verifiers with anonymous proof of the system environment and configuration, thereby ensuring the legitimacy of the system. Another application is associated with smart cards. For cases where the credential issuer and verifier are the same entity, Chase *et al.* [6] propose keyed-verification anonymous credentials (KVAC), which utilizes algebraic message authentication codes to enhance the efficiency of credential issuance and verification. Later, Camenisch *et al.* [7] combine KVAC with BBS signatures, further increasing the efficiency of hbAC (KVAC on smart cards). In a general setting, Bichsel *et al.* [8] consider the implementation of the CL-credentials [9] on Java cards. However, to maintain the independence of the smart card in credential usage, all computations must be completed within the smart card, which results in low computational efficiency under practical security parameters. Hanzlik *et al.* [3] focus on offloading computational overhead to the devices and decoupling the smart card's computational burden from the number of attributes, thereby improving overall performance.

Tracing misbehaved credential users. The commonly used ACs, such as U-Prove, Idemix, and DAA, have introduced

TABLE 1: Asymptotic complexities of different AC systems (traceability part only).

Scheme	Prove	Verify	Trace			|Evidence|	Approach
			type (a)	type (b)	type (c)		
[9]	$O(1)$	$O(1)$	$\mathbf{O(1)}$	$O(L)$	$O(L)$	$O(1)$	T-PKE
[10]	$O(1)$	$O(1)$	$\mathbf{O(1)}$	$O(L)$	$O(L)$	$O(1)$	T-PKE
[11]	$O(1)$	$O(1)$	$O(S)$	$O(S+L)$	$O(S+L)$	$\mathbf{0}$	T-PKE
[12]	$O(1)$	$O(1)$	$\mathbf{O(1)}$	$O(L)$	$O(L)$	$O(1)$	T-PKE
[13]	$O(1)$	$O(1)$	$O(S)$	$O(S+L)$	$O(S+L)$	$\mathbf{0}$	T-BD
[14]	$\mathbf{0}$	$\mathbf{0}$	$O(S)$	$O(L)$	$O(S+L)$	$\mathbf{0}$	T-BD
This work	$O(1)$	$O(1)$	$\mathbf{O(1)}$	$\mathbf{O(R+L)}$	$\mathbf{O(R+L)}$	$O(1)$	T-PKE

new issues such as rampant malicious behavior, as ACs allow users to perform any malicious action after anonymous authentication. The privacy features of ACs make it impossible to trace wrongdoers [10]. A straightforward solution is introducing tracing authorities who own tracing keys and are capable of linking the anonymous owners behind the credentials. This solves the problem of traceability and enhances the security level and stability of the system using ACs. Together with the hardware/device setting, it is almost impossible for wrongdoers to dodge the tracing. However, the aforementioned AC constructions did not put these things together. Thus the feasibility and efficiency of such a combination are still not clear.

Drawbacks of existing schemes. The notion of traceability of anonymous credentials comes from the vein of group signatures and is attracting increasing attention in recent works. Existing traceable anonymous credential schemes can briefly be categorized into two approaches: public key encryption-based (T-PKE) [9], [10], [11], [12] and backdoor-based (T-BD) [13], [14]. The former requires a regulator to select a verifiable encryption scheme and release the encryption public key, so users can encryption their identity during each authentication for identity tracing. In contrast, backdoor-based schemes will not introduce additional operations on the user side, as the user-specific backdoor was generated based on his private key. The main drawback of both approaches is that they typically focus on "de-anonymizing" an anonymous malicious user based on his authentication records. Due to this reason, the capability of current traceable AC is somewhat limited and struggles to trace anonymous users on larger scales, such as auditing the authentication records across service providers or identifying repeated authentication of the same user [15]. The latter can be viewed as an essential security property. For example, in the well-known anonymous voting system Helios and its improvements [16], [17], [18], user privacy and the validity of the voting can both benefit from de-duplicating the votes from the same user.

We compare some related works with our work in Table 1, where S is the total number of registered users, L is the number of valid authentication records, and R is the number of different authentication rounds. Then, **|Evidence|** denotes the size for additional messages for traceability, while the other fields represent computational complexity. The

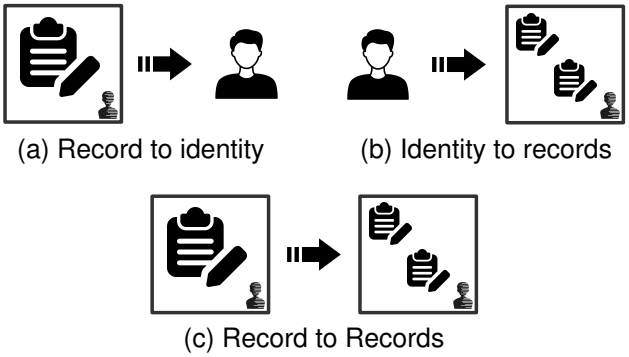

Figure 1: Different types of identity tracing.

On the complexity of tracing. From the regulator's perspective, identity tracing tasks can be categorized into three types (as shown in Figure 1): (a) identifying the credential user corresponding to a particular authentication record; (b) tracing other authentication records linked to a specific user; and (c) identifying other authentication records associated with a given authentication record. Type (c) can be seen as a combination of type (a) and type (b). Let S and L denote the total number of registered users and their authentication records maintained by the service providers, respectively. For type (a), only one decryption will be needed by T-PKE, whereas the T-BD scheme typically requires approximately $S/2$ rounds of matching calculations to complete. For type (b), both tracking schemes demand L rounds of computation. Notably, the T-PKE scheme must ensure user identity privacy throughout the computation process; otherwise, the unlinkability between unrelated users and their respective authentication records may be compromised. For type (c), both schemes require safeguarding user identity privacy during computation. The T-PKE scheme demands $L+1$ rounds of decryption, while the T-BD scheme averages $S/2 + L$ rounds of computation.

To ensure user privacy, the regulators' private keys must be protected from leakage, and service providers must be assured that the certification records they maintain are not misused. To alleviate the computational burden on regulators, advanced but time-consuming cryptographic primitives (such as secure multi-party computation or private set intersection) could be employed to enable outsourced compu-

tation. Notably, real-world scenarios may involve multiple trust domains, various credential issuers, service providers, and regulators, which make type (b) and (c) identity tracing tasks complicated. Thus, enhancing regulatory efficiency remains a critical and pressing challenge.

Our goals and setting. To overcome the aforementioned problems, we consider designing a new tracing scheme compatible with the hardware-binding AC systems [3], [4]. We consider designing an improved encryption scheme with special functionalities that can be naturally integrated into this setting as a plugin, which is of independent interest. Intuitively, tracing malicious users (especially type (b) and type (c)) entails unavoidable overheads, such as iterating all authentication records to identify those linked to the target user. Privacy concerns and computational offloading will introduce additional costs, which make the tracing process more complicated.

An intuitive solution is searchable encryption, which supports keyword searches on encrypted data. However, searchable symmetric encryption (SSE) is limited to a single-user model and is unsuitable for authentication scenarios. Public key encryption with keyword search (PEKS), on the other hand, manages backdoors for every keyword and uses the corresponding one for each search. However, the hash functions used by PEKS when generating backdoor and ciphertext bring potential difficulties to the verifiability proof of encrypted content. The keywords in PEKS can be made public, but the user identity, with the same status as the keyword, is private information. Therefore, introducing searchable encryption to traceable AC is still challenging.

We also observe that most widely used AC systems (such as U-Prove [19], Idemix [20], and BBS signatures [21], [22]) generally have algebraic structures similar to Pedersen Commitment. To use the credential, users must prove the knowledge of a private element to demonstrate ownership of the credential. This element is typically considered the user's private key and can be naturally stored in trusted hardware to achieve credential separation. Besides, recent credentials with different building blocks, such as schemes based on aggregatable signatures [14] and SPS-EQ [23], [24], advancements in functionality and efficiency. While these schemes introduce new features, they also alter the process, as the way how the private key will be used is changed. This presents challenges for migrating such credentials into hbAC, as well as for the corresponding tracing scheme, which is one of the main issues we aim to address in the future.

High-level overview of our approach. Here we provide a high-level overview of our tracing paradigm (see Figure 2). Initially, the user can obtain credentials from the issuer based on hardware with the user's secret key, and then store them on the device. For hardware matching, we mean... For each authentication, the device has to cooperate with the hardware to achieve: 1) proof of possession of the credentials (as well as the requested attributes); and 2) generate proof of regulatory text of authentication records. During

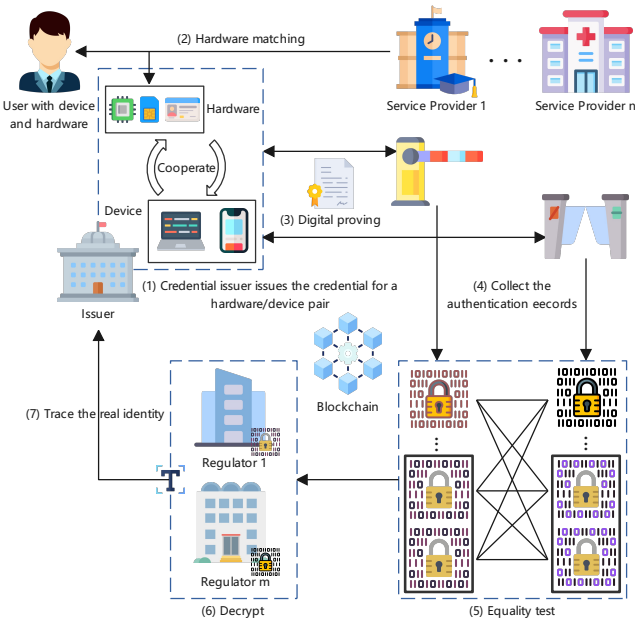

Figure 2: High-level overview of our approach.

this procedure, the hardware only produces a partial token but ensures the secret key will never be reconstructed outside the hardware. The device then completes the whole showing token and sends it to the verifier who either accepts or rejects it. Combining them ensures that the information recorded in the regulatory text is bound to the trusted hardware that dominates the credentials. To trace an anonymous user, both regulators and service providers can obtain clues by comparing the regulatory texts (generated at different authentication rounds), and finally de-anonymizing the user. Regulators can also achieve privacy-preserving cross-domain identity tracing by creating and distributing equality test backdoors for specific users.

Related works about traceable anonymous credentials. We reviewed recent works about credential tracing [10], [12], [14]. Hébant *et al.* [14] propose a tracing method for ACs with randomizable public keys. The tracing key for a user can be treated as another public key of him (on another group of the bilinear pair). Kohlweiss *et al.* [10] design a privacy-preserving blueprint scheme supporting arbitrary function relations f based on a new homomorphic encryption primitive they proposed, allowing the regulators to trace all escrowed user data that meets the relation f. However, the main focus of the blueprint is to limit the tracing ability of regulators to protect the privacy of other users, which makes it less efficient. Wang *et al.* [12] use an encryption-based approach, allowing users to generate pseudonyms after each authentication. For tracing, the regulator decrypts and obtains the user-specific backdoor, which can help produce all potential pseudonyms for that user (depending on the number of pseudonyms in the system).

In [5], Brickell, Camenisch, and Chen employ a spe-

cialized chip and construct an anonymous authentication scheme (DAA), although not strictly an AC system. DAA provides a selective linkability of the pseudonyms if one of the public parameters of the verifier remains identical. Although such a mechanism is useful for detecting rogue members under some circumstances, it breaks the security requirement of ACs and is unable to trace the real identity of the user. In [25], Baldimtsi and Lysyanskaya introduce a lightweight credential system that supports user tracing for illegal use, but only valid for single-use credentials.

In [11], Canard and Lescuyer use ElGamal encryption to generate regulatory key pairs during the initialization phase (though this is not explicitly stated in the paper). Later, the regulator can decrypt and obtain the re-randomized user public key, and trace the real identity by comparing it with other sanitizers' public keys. We follow a similar path but with different cryptography primitives–the probabilistic public key encryption (PKEET) proposed by Yang *et al.* [26] that allows anyone to compare two ciphertexts and check if they are encryptions of the same message. However, the corresponding schemes rely on the hash functions as the main building block, which does not support verifiable encryption. We solve this issue by introducing a new encryption scheme for better tracing efficiency and compatibility with hbAC.

1.1. Our Contributions and Technical Overview

Our contribution in this paper can be summarized in points as follows:

Verifiable public key encryption with equality test. We formalize a cryptographic primitive called twisted ElGamal encryption with equality test (TEET). The key observation is that twisted ElGamal encryption [27] is zero-knowledge proof friendly and PKEET [26] supports equality test ciphertexts regardless of the public keys. However, to achieve the desired security level (OW-CCA2 security), the latter relies on the hash function, which will bring difficulties to zero-knowledge proof and their combination with hbAC. Although recent studies show that the overhead of SNARK proofs can be significantly reduced by using new hash functions [28] or outsourcing them [29], how hardware with limited capabilities can compute such a proof of encrypted content with other device remains underexplored. TEET inherits the advantages of both schemes mentioned above – the verifiability of ciphertexts and their equality test. This makes the verifier convinced during the authentication process that the regulatory text contains the user's long-term identity. Meanwhile, during the tracing of authentication records, the user's identity privacy within the regulatory text is still protected. We should mention that in contrast to standard construction, our scheme achieves only OW-CPA security under CDH assumptions. However, the strong Fiat-Shamir transformation [30] can turn TEET into a non-malleable one, which is secure enough to satisfy our design goals.

Round isolation for equality test. We focus on the privacy security of PKEET, which was raised by the one-wayness (due to the equality test) of the original scheme. When considering anonymous authentication and regulation, this property poses a privacy risk to users, as each regulatory text generated during authentications can be used to match historical records. Our solution is to introduce a new unknown order generator for different rounds of authentications (e.g. period), thereby restricting the capability of the equality test to specific rounds. Such type of generators can be selected by a trusted third party and assigned with proof of reliability, we won't further discuss this part in this paper. With this modification, PKEET achieves perfect soundness within rounds and computational soundness across rounds (see Section 3.3). To trace identity across different service providers (or different rounds), regulators can use this property to generate matching texts to outsource tracking tasks to service providers securely.

Efficient traceable hbAC instantiation. Our hbAC builds upon the AC system with the card/helper settings by Hesse *et al.* [4] but we add the round-specific generator and TEET as a plugin. We instantiate our scheme in both type-1 bilinear groups $\mathsf{BG}_1 = (p, \mathbb{G}_1, g, \mathbb{G}_T, e)$ with pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, and type-3 bilinear groups $\mathsf{BG}_3 = (p, \dot{\mathbb{G}}_1, \bar{g}, \dot{\mathbb{G}}_2, \tilde{g}, \dot{\mathbb{G}}_T, \dot{e})$ with pairing $\dot{e} : \dot{\mathbb{G}}_1 \times \dot{\mathbb{G}}_2 \to \dot{\mathbb{G}}_T$. The reason is that TEET encryption and the BBS+ signature rely on different types of pairing. Thus, we cannot simply reuse some variables in both components to maintain consistency between them. Therefore, we show how to securely and efficiently prove this relation: we reuse the random number that was securely shared between the hardware and the device via a PRF key held by both parties, then conduct a zero-knowledge proof of the variables on different groups ($\mathbb{G}_1$ and $\dot{\mathbb{G}}_1$). This allows us to prove the consistency without introducing additional rounds of interaction. To ensure the security of the message output by the hardware when communicating with the device, all of them were masked by the PRF output.

2. Preliminaries and Notation

2.1. Bilinear Groups

Bilinear groups are a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of order p along with a map, called pairing, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that is:

- bilinear: for any $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$;
- non-degenerate: for any $g \neq 1_{\mathbb{G}_1}$ and $\tilde{g} \neq 1_{\mathbb{G}_2}$, $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$;
- efficient: for any $g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$, $e(g^a, g^b)$ can be efficiently computed.

We follow the definition in [31] of the three types of pairings: in type 1, $\mathbb{G}_1 = \mathbb{G}_2$; in type 2, $\mathbb{G}_1 \neq \mathbb{G}_2$ but efficient homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ exists, while no efficient one exists in the other direction; in type 3, $\mathbb{G}_1 \neq \mathbb{G}_2$

no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$, in either direction.

Definition 1 (Bilinear group generator). A bilinear group generator BGGen is a (possibly probabilistic) polynomial-time algorithm that takes as input a security parameter 1^λ and outputs a description of a bilinear group $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$ consisting of groups $\mathbb{G}_1 = \langle g \rangle$, $\mathbb{G}_2 = \langle \tilde{g} \rangle$, and $\mathbb{G}_T$ of prime order p with $\lceil \log_2 p \rceil = \lambda$ and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

2.2. Computational Assumptions

- Discrete Logarithm (DL) assumption: Given $(g, g^a) \in \mathbb{G}^2$, the DL assumption in the group $\mathbb{G}$ states that there is no probabilistic polynomial time ($\mathcal{PPT}$) algorithm that can recover a with nonnegligible advantage.
- Decisional Diffie–Hellman (DDH) assumption: Given $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, the DDH assumption in the group $\mathbb{G}$ states that there is no $\mathcal{PPT}$ algorithm that can decide whether $c = a \cdot b$ or c is random with nonnegligible advantage.
- q-Strong Diffie– Hellman (q-SDH) assumption: Given a (q+2) tuple $(g, g^x, g^{x^2}, \ldots, g^{x^q})$, the q-SDH assumption in the group $\mathbb{G}$ states that there is no $\mathcal{PPT}$ algorithm that can output a pair $(c, g^{\frac{1}{x+c}})$ where $c \in \mathbb{Z}_p$ with nonnegligible advantage.

2.3. Zero-Knowledge Protocols

Zero-knowledge protocol enables a prover to convince a verifier that a statement is true without revealing anything except the validity of the statement. An interactive zero-knowledge proof system is called a Sigma protocol (Σ-protocol) if it contains 3 phases between the prover P and the verifier V as below:

- (Commit) P sends a first message a to V;
- (Challenge) V sends a random challenge e to P;
- (Response) P replies with a second message z.

For any NP relation $(x, \omega) \in R$, where ω is the witness of the statement x, a valid Sigma protocol is required to satisfy standard completeness and the variants of soundness and zero-knowledge as below:

- Completeness. If P and V follow the protocol, then V always accepts.
- Special soundness. For any $x \in X$ and any pair of accepting transcripts (a, c, r), (a, c', r') with $c \neq c'$, there exists a $\mathcal{PPT}$ extractor outputs a witness ω for x.
- Special honest-verifier zero-knowledge (SHVZK). There exists a $\mathcal{PPT}$ simulator $\mathcal{S}$ such that for any $x \in X$ and challenge c, $\mathcal{S}$ produces conversations (a, c, r) with the same probability distribution as conversations between honest P and V.

We also use Signature of Knowledge (SoK) as non-interactive zero-knowledge proof (following the definition in [32] and [33]) while designing the authentication protocol.

2.4. Card-based Anonymous Credential with BBS+ Signature

A card-based Anonymous Credential (cbAC) system [4] is an anonymous credential scheme with visual holder authentication. It contains three interactive procedures (Setup,Join,$\mathsf{Present}$) which will be executed between an issuer, and arbitrary tamper-resistant smartcards, holders, and verifiers. In their settings, the verifier can visually verify that the picture on the smartcard (hereinafter referred to as card) matches the individual. The holder must cooperate with the card to produce valid proof of knowledge of the selectively disclosed attributes and the corresponding BBS+ signature. Both of them will receive the shared state from a trusted card issuer $\mathcal{F}_{\mathsf{cardAuth}}$, consisting of a Pseudo-Random Function (PRF) key K and a non-hiding commitment $Q = h_1^m$ to attribute m contributed by the card.

We briefly review the standard BBS+ signature: Let $g, h_1, \ldots, h_\ell \in \dot{\mathbb{G}}_1^{\ell+1}$ and $\tilde{g}, \tilde{h}_1, \ldots, \tilde{h}_n \in \dot{\mathbb{G}}_2^{n+1}$ be the generators. The issuer randomly chooses $\gamma \leftarrow\!\!\$\ \mathbb{Z}_p^* (\stackrel{\mathsf{def}}{=} \mathbb{Z}_p \backslash \{1\})$ and set $(\gamma, w = \tilde{g}^\gamma)$ as the secret-public key pair. Given messages $\boldsymbol{m} = (m_1, \ldots, m_\ell) \in \mathbb{Z}_p$, the issuer randomly chooses $e, s \leftarrow\!\!\$\ \mathbb{Z}_p^2$ and computes $A = (g g^s \prod_{i=1}^{\ell} h_i^{m_i})^{\frac{1}{e+\gamma}}$. The BBS+ signature can be verified by checking whether $e(A, w\tilde{g}^e) = e(g g^s \prod_{i=1}^{\ell} h_i^{m_i}, \tilde{g})$ holds. BBS+ signature satisfies the EUF-CMA security if the q-SDH problem is hard in the bilinear group [34].

2.5. Twisted ElGamal Encryption

Twisted ElGamal [27] is modified from the standard ElGamal encryption algorithm, it switched the roles of key encapsulation and session key, and lifted the message m on a new generator. Its ciphertext has the same structure as Pedersen commitment, so it can easily connect with existing zero-knowledge proof systems (such as sigma protocol [35]). Twisted ElGamal is IND-CPA secure under the DDH assumption. We recall the algorithm $\Pi_{\mathsf{TE}} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ proposed in [27] as follows:

- $\mathsf{Setup}(1^\lambda)$: Run $(\mathbb{G}_1, g, p) \leftarrow \mathsf{GroupGen}(1^\lambda)$, pick $h_1 \leftarrow\!\!\$\ \mathbb{G}_1^* (\stackrel{\mathsf{def}}{=} \mathbb{G}_1 \backslash \{1\})$, set $pp = (\mathbb{G}_1, g, h_1, p)$ as global public parameters. The randomness and message spaces are $\mathbb{Z}_p$.
- $\mathcal{G}(1^\lambda)$: On input pp, choose $sk \leftarrow\!\!\$\ \mathbb{Z}_p$, set $pk = g^{sk}$.
- $\mathcal{E}(pk, m)$: Pick $r \leftarrow\!\!\$\ \mathbb{Z}_p$,compute $X = pk^r$, $Y = g^r h_1^m$, output $\mathsf{C} = (X, Y)$.
- $\mathcal{D}(sk, C)$: Parse $\mathsf{C} = (X, Y)$, compute $h_1^m = Y / X^{sk^{-1}}$, recover m from h_1^m.

2.6. Public-Key Encryption with Equality Test

Also denoted as PKEET, is a primitive proposed by Yang *et al.* [26] that can categorize ciphertexts, even those

encrypted with different public keys, with the same underlying messages into one cluster. We recall the algorithm $\Pi_{\mathsf{PKEET}} = (\mathcal{G}, \mathcal{E}, \mathcal{D}, \mathsf{T})$ proposed in [26] as follows:

- $\mathcal{G}(1^\lambda)$: Pick $x \leftarrow\$ \mathbb{Z}_p$ and compute $y = g^x$. Set $pk = y$ and $sk = x$.
- $\mathcal{E}(m, y)$: Let $m \in \mathbb{G}_1^*$, pick $r \leftarrow\$ \mathbb{Z}_p^*$, compute $U = g^r$, $V = m^r$, $W = \mathsf{H}(U, V, y^r) \oplus m||r$. The ciphertext is $\mathsf{C} = (U, V, W)$.
- $\mathcal{D}(C, x)$: To decrypt $\mathsf{C} = (U, V, W)$, compute $m||r \leftarrow \mathsf{H}(U, V, U^x) \oplus W$. If $(m \in \mathbb{G}_1^* \wedge r \in \mathbb{Z}_p^* \wedge U = g^r \wedge V = m^r)$, return m; otherwise, return $\perp$.
- $\mathsf{T}(\mathsf{C}_1, \mathsf{C}_2)$: Given two ciphertexts $\mathsf{C}_1 = (\mathsf{C}_1, V_1, W_1)$ and $\mathsf{C}_2 = (U_2, V_2, W_2)$, test if $e(U_1, V_2) = e(U_2, V_1)$, return 1; otherwise, return 0.

3. Twisted ElGamal Encryption with Equality Test

At a high-level overview, we need a construction that supports the hiding of sensitive information while supporting the equivalence comparison. The first requirement ensures that the users can only selectively disclose the required attributes, while the second gives the service provider the ability to de-duplicate the users according to their authentication information. Specifically, we use twisted ElGamal to better adapt to the BBS+ signature based cbAC systems without introducing excessive knowledge proofs for consistency between them.

Now, we extend the capabilities of the Twisted ElGamal (TE) encryption algorithm by combining it with PKEET to get Twisted ElGamal with Equality Test (TEET). This allows the consistency of the plaintext to be checked without decrypting the corresponding ciphertext. We use a non-hiding commitment as the input and output of the encryption and decryption methods of TEET.

3.1. Definition

Definition 2 (TEET). A *twisted ElGamal encryption with equality test* is a verifiable public key encryption scheme $\Pi_{\mathsf{TEET}} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ of the following polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda)$ is a probabilistic algorithm which call $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$, pick two random generators in $\mathbb{G}_1$, and return them as public parameter pp.
- $\mathcal{G}(\lambda)$ is a probabilistic algorithm which on input the public parameter pp and outputs the decryption and encryption key pair (sk, pk).
- $\mathcal{E}(pk, Q)$ is a probabilistic algorithm which on input an encryption public key $pk \in \mathbb{G}_1$, a message $Q \in \mathbb{G}_1^*$ ($\stackrel{\mathsf{def}}{=} \mathbb{G}_1 \backslash \{1\}$), outputs a ciphertext C.
- $\mathcal{D}(sk, \mathsf{C})$ is a deterministic algorithm which given a decryption secret key $sk \in \mathbb{Z}_p$ and a ciphertext C, outputs the message Q.
- $\mathsf{Test}(\mathsf{C}, \mathsf{C}')$ is a deterministic algorithm which given two valid ciphertexts C and C' returns 1 on equality of plaintext (under the same round) and 0 otherwise.

The correctness requirement is that $\forall\ \lambda \in \mathbb{N}$ and $\forall\ Q \in \mathsf{PtSp}(\lambda)$, $(pk, sk) \leftarrow \mathcal{G}(1^\lambda), Q \leftarrow \mathcal{D}(sk, \mathcal{E}(pk, Q))$ where $\mathsf{PtSp}(\lambda)$ is the message space associated to TEET.

We say TEET has *Ciphertext Comparability* with error ϵ for some function $\epsilon(\cdot)$ if there exists an efficiently computable deterministic function $\mathsf{Test}(\mathsf{C}, \mathsf{C}')$ such that for every λ we have:

1) Perfect Consistency: for every $x \in \mathsf{PtSp}(\lambda)$

$$\Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathcal{G}(1^\lambda), (pk', sk') \leftarrow \mathcal{G}(1^\lambda), \mathsf{C} \leftarrow \mathcal{E}(pk, x) \\ \mathsf{C}' \leftarrow \mathcal{E}(pk', x) : \mathsf{Test}(\mathsf{C}, \mathsf{C}') = 1\end{array}\right] = 1$$

2) Soundness: for every polynomial time algorithm $\mathcal{M}$

$$\Pr\left[\begin{array}{l}(\mathsf{C}, \mathsf{C}', sk, sk') \leftarrow \mathcal{M}(1^\lambda), x \leftarrow \mathcal{D}(sk, \mathsf{C}), \\ x' \leftarrow \mathcal{D}(sk', \mathsf{C}') : x \neq \perp \wedge x' \neq \perp \wedge x \neq x' \wedge \\ \mathsf{Test}(\mathsf{C}, \mathsf{C}') = 1\end{array}\right] \leq \epsilon(\lambda)$$

In the above definition, consistency ensures that encryptions (even under different public keys) of the same message with the same round-specific generator can be recognized. We also define soundness to measure the probability of false-hits (i.e. $\mathsf{Test}(\mathsf{C}, \mathsf{C}') = 1$ but C and C' are encryptions of different messages).

Definition 3 (OW-CPA). We followed the definition of OW-CPA in [26]. Let $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a public key encryption scheme and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a polynomial-time adversary. For $\lambda \in \mathbb{N}$ let

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathcal{A},\Pi} \stackrel{\mathsf{def}}{=} \Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathcal{G}(1^\lambda), \delta \leftarrow \mathcal{A}_1^{\mathcal{O}_\mathcal{E}}(pk) \\ x \leftarrow \mathsf{PtSp}(\lambda), y \leftarrow \mathcal{E}(pk, x) \\ x' \leftarrow \mathcal{A}_2^{\mathcal{O}_\mathcal{E}}(pk, \delta, y) : x' = x\end{array}\right]$$

We say that Π is secure in OW-CPA if $\mathsf{Adv}^{\text{ow-cpa}}_{\mathcal{A},\Pi}$ is negligible for any $\mathcal{A}$.

3.2. Construction

- $\mathsf{TEET.Setup}(1^\lambda)$: Run $\mathcal{G}_1 = (p, \mathbb{G}_1, \mathbb{G}_T, e, g) \leftarrow \mathsf{BGGen}_1(1^\lambda)$, $(h, h_1) \leftarrow\$ \mathbb{G}_1^2$, and set $pp = (p, \mathbb{G}_1, \mathbb{G}_T, e, g, h, h_1)$ as global public parameters. Let the randomness space be $\mathbb{Z}_p$ and the message space be $\mathbb{G}_1^*$.
- $\mathsf{TEET}.\mathcal{G}(1^\lambda)$: Choose $sk \leftarrow\$ \mathbb{Z}_p$, set $pk = g^{sk}$.
- $\mathsf{TEET}.\mathcal{E}(pk, Q; r, v)$: Compute $X = pk^r$, $Y = g^r Q$, $U = Q^v$, $K = h^v$, output $\mathsf{C} = (X, Y, U, K)$.
- $\mathsf{TEET}.\mathcal{D}(sk, \mathsf{C})$: Parse $\mathsf{C} = (X, Y, U, K)$, compute $Q = Y / X^{sk^{-1}}$, and check if $e(U, h) = e(Q, K)$ holds. If so, output Q, otherwise output $\perp$.
- $\mathsf{TEET.Test}(\mathsf{C}, \mathsf{C}')$: Given two ciphertexts $\mathsf{C} = (X_1, Y_1, U_1, K_1)$ and $\mathsf{C}' = (X_2, Y_2, U_2, K_2)$, if $e(U_1, K_2) = e(U_2, K_1)$, return 1; otherwise, return 0.

Theorem 1. The PKE scheme Π_{TEET} with message space $\mathbb{G}_1^*$ is OW-CPA secure based on the CDH assumption.

Proof 1.

Let $\mathcal{A}$ be a PPT adversary attacking the OW-CPA security of the above PKE scheme. Suppose that $\mathcal{A}$ runs in time t and makes at most q_E encryption queries. Let $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}CPA}}(t, q_E)$ denote the advantage of $\mathcal{A}$ in the OW-CPA experiment.

It remains to prove $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}CPA}}(t, q_E)$ is negligible. We prove this by showing if not so, we can build an adversary $\mathcal{B}$ and break the CDH assumption with the same advantage. Let the public parameter be $(p, \mathbb{G}_1, \mathbb{G}_T, e, g)$, given the CDH challenge instance (g, g^a, g^b), $\mathcal{B}$ is asked to compute g^{ab}. To do so, $\mathcal{B}$ interacts with $\mathcal{A}$ by simulating $\mathcal{A}$'s observation in the following experiment.

1) Setup: $\mathcal{B}$ picks $t \leftarrow\$ \mathbb{Z}_p$, sends $pp = (p, \mathbb{G}_1, \mathbb{G}_T, e, g^a)$ and $pk = g$ and $h = (g^a)^t = g^{at}$ to $\mathcal{A}$. Here, g^a, g, and g^{at} serve as the original generator of $\mathbb{G}_1$, encryption public key, and another generator in $\mathbb{G}_1$ respectively.
2) Challenge: $\mathcal{B}$ selects $(u, v) \leftarrow\$ \mathbb{Z}_p^2$, sends $\mathsf{C} = (X = g^b, Y = g^{au}, U = (\frac{g^u}{g^b})^v, K = (g^t)^v)$ to $\mathcal{A}$.
3) Query: $\mathcal{A}$ sets $T = \emptyset$ and makes at most q_E encryption queries: selects $\mathfrak{r}, \mathfrak{v} \leftarrow\$ \mathbb{Z}_p^2$ and $Q^* \leftarrow\$ \mathbb{G}_1^*$, computes $X^* = pk^{\mathfrak{r}} = g^{\mathfrak{r}}$, $Y^* = g^{a\mathfrak{r}}Q^*$, $U^* = Q^{*\mathfrak{v}}$, $K^* = h^{\mathfrak{v}} = g^{at\mathfrak{v}}$. Let $T = T \cup \{(X^*, Y^*, U^*, K^*, Q^*)\}$.
4) Guess: When $\mathcal{A}$ outputs Q^*, $\mathcal{B}$ outputs Y/Q^*.

It is obvious that $\mathcal{B}$'s simulation is perfect. If $\mathcal{A}$ outputs Q^*, it means that one of the following two events occurred:

1) **Event $\mathbf{E_0}$**: If there is an entry $(X^*, Y^*, U^*, K^*, Q^*)$ in the set T such that $e(U^*, K) = e(U, K^*)$, return Q^*. Then we have

$$\begin{aligned} e(U^*, K) &= e(U, K^*) \\ e(Q^{*\mathfrak{v}}, g^{tv}) &= e(g, g)^{v(u-b)at\mathfrak{v}} \\ Q^* &= g^{a(u-b)} \\ Y/Q^* &= \frac{g^{au}}{g^{a(u-b)}} = g^{ab} \end{aligned}$$

Obviously, $\Pr[\mathbf{E_0}]$ is negligible because Q^* was randomly selected in $\mathbb{G}_1^*$. So we have

$$\Pr[\mathbf{E_0}] \leq \frac{q_E}{2^\lambda}$$

2) **Event $\mathbf{E_1}$**: Compute Q^* with instance (pp, pk, h, C). If $e(U, h) = e(Q^*, K)$ holds, return Q^*. Then we have

$$\begin{aligned} e(U, h) &= e(Q^*, K) \\ e(Q^*, g^{tv}) &= e(g, g)^{atv(u-b)} \\ Q^* &= g^{a(u-b)} \\ Y/Q^* &= \frac{g^{au}}{g^{a(u-b)}} = g^{ab} \end{aligned}$$

Therefore, we have

$$\begin{aligned} \mathsf{Adv}_{\mathcal{B}}^{\mathsf{CDH}} \geq \Pr[\mathbf{E_1}] &= \mathsf{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}CPA}}(t, q_E) - \Pr[\mathbf{E_0}] \\ &\geq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{OW\text{-}CPA}}(t, q_E) - \frac{q_E}{2^\lambda} \end{aligned}$$

Thus $\mathcal{B}$ can break the CDH assumption with a non-negligible advantage, which results in a contradiction. Putting all the above together, Theorem 1 follows.

3.3. Provable Security

The randomnesses used in TEET are identical between (X, Y) and (U, K), while the former can be treated as a TE ciphertext. Inspired by the work of Bernhard *et al.* [30] on the legitimacy of ballot casting, we add additional proof to ensure the same message Q was used in Y and U (without disclosing the value of Q). So that the equality of different TEET ciphertexts can only be tested with the same approach of PKEET. This can help us to integrate other researches related to PKEET, such as authorization policies [36] or scenarios of outsourced computation [37], to enhance the functionality and expand application scenarios.

We define the relation mentioned above as L_{valid}, and let $r_1 = -rv$. We further require the prove of knowledge of m (assuming Q was calculated by $Q = h_1^m$) for specific consideration (see Section xx):

$$\begin{aligned} L_{valid} = \{(pk, g, h, h_1, \mathsf{C} = (X, Y, U, K)) | \exists\ r, m, v, r_1\ s.t. \\ X = pk^r \wedge \\ Y = g^r h_1^m \wedge \\ U = K^m \wedge \\ U = Y^v g^{r_1}\} \end{aligned}$$

Sigma protocol for L_{valid}. To prove L_{valid} in zore knowledge, we design a Sigma protocol $\Sigma_{valid} = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for L_{valid} to prove that the twisted ElGamal encryption ciphertext (X, Y) and the corresponding pair for equality test (U, K) consist of the same value $Q = h_1^m$. The Setup algorithm of Σ_{valid} is the same as that of the twisted ElGamal. On statement (g, h, h_1, X, Y, U, K), P and V interact as below:

1. P picks $a, b, c, d \leftarrow\$ \mathbb{Z}_p^4$, sends $A = pk^a$, $B = g^a h_1^b$, $C = K^b$, $D = Y^c g^d$ to V.
2. V picks $e \leftarrow\$ \mathbb{Z}_p$ and sends it to P as the challenge.
3. P computes $z_1 = a + er$, $z_2 = b + em$, $z_3 = c + ev$, $z_4 = d + er_1$ using witness $w = (r, m_1, v, r_1)$, then sends (z_1, z_2, z_3, z_4) to V. V accepts if and only if the following three equations hold simultaneously:

$$pk^{z_1} = AX^e \tag{1}$$

$$g^{z_1} h_1^{z_2} = BY^e \tag{2}$$

$$K^{z_2} = C \cdot U^e \tag{3}$$

$$Y^{z_3} g^{z_4} = DU^e \tag{4}$$

Lemma 1. Σ_{valid} is a public-coin SHVZK proof of knowledge for L_{valid}.

Proof 2. We prove that all three properties required for Σ_{valid} are met.

Perfect completeness is obvious from a simple calculation.

To show *special soundness*, we fix the initial message (A, B, C, D), suppose there are two accepting transcripts

(e, z_1, z_2, z_3, z_4) and $(e', z'_1, z'_2, z'_3, z'_4)$ with $e \neq e'$, the witness can be extracted as below. From (1), we have $z_1 = a + er$ and $z'_1 = a + e'r$, which implies $r = (z_1 - z'_1)/(e - e')$. And so as from (2), (3) and (4), we can compute $m = (z_2 - z'_2)/(e - e')$, $v = (z_3 - z'_3)/(e - e')$ and $r_1 = (z_4 - z'_4)/(e - e')$.

To show *special HVZK*, for a fixed challenge e, the simulator $\mathcal{S}$ works as below: picks $z_1, z_2, z_3, z_4 \leftarrow\!\$\ \mathbb{Z}_p^4$, then computes $A^* = pk^{z_1}/X^e$, $B^* = g^{z_1} h_1^{z_2}/Y^e$, $C^* = K^{z_2}/U^e$, $D^* = Y^{z_3} g^{z_4}/U^e$. Obviously, $(A^*, B^*, C^*, D^*, e, z_1, z_2, z_3, z_4)$ is an accepting transcript, and it is distributed as in the real protocol.

This proves Lemma 1.

According to the result in [30], let sFS be a strong Fiat-Shamir transformation, and Σ be a sigma protocol with a challenge space that is exponentially large in the security parameter. Then $\mathsf{sFS}.\Sigma$ is *zero-knowledge* and *simulation sound extractable* with respect to expected polynomial-time adversaries.

Theorem 2. The above PKE scheme has perfect consistency, perfect soundness-SR, and computational soundness-DR (whether the h used in $\mathcal{E}$ is the same or not).

Proof 3. The proof is straightforward, as follows:

1. Perfect Consistency. For any $(pk, sk) \leftarrow \mathcal{G}(1^k)$, $(pk', sk') \leftarrow \mathcal{G}(1^k)$ and $\mathsf{C} \leftarrow \mathcal{E}(pk, m, h)$, $\mathsf{C}' \leftarrow \mathcal{E}(pk', m, h)$ where $\mathsf{C} = (X, Y, Q^v, h^v)$, $\mathsf{C}' = (X', Y', Q^{v'}, h^{v'})$, we have
$$e(Q^v, h^{v'}) = e(Q^{v'}, h^v) = e(Q, h)^{vv'}$$
for any $Q = h_1^m$ ($m \in \mathbb{Z}_p^*$) and $(v, v') \in \mathbb{Z}_p^{*2}$.
2. Perfect Soundness-SR. Given two ciphertexts $\mathsf{C} = (X, Y, Q^v, h^v)$, $\mathsf{C}' = (X', Y', Q'^{v'}, h^{v'})$, we have
$$e(Q^v, h^{v'}) = e(Q, h)^{vv'}, e(Q'^{v'}, h^v) = e(Q', h)^{vv'}$$
then it must be true that $e(Q, h)^{vv'} \neq e(Q', h)^{vv'}$ for any $Q \neq Q'$ ($m \neq m'$) and $(v, v') \in \mathbb{Z}_q^{*2}$.
3. Computational Soundness-DR. Given two ciphertexts $\mathsf{C} = (X, Y, Q^v, h_1^v)$ and $\mathsf{C}' = (X', Y', Q'^{v'}, h_2^{v'})$ encrypted under any pair of different unknown order generators $(h_1, h_2) \in \mathbb{G}_2^{*2}$ ($h_1 \neq h_2$), we have
$$e(Q^v, h_1^{v'}) = e(Q, h_1)^{vv'}, e(Q'^{v'}, h_2^v) = e(Q', h_2)^{vv'}$$
then it must be true that $e(Q, h_1)^{vv'} \neq e(Q', h_2)^{vv'}$ for $(v, v') \in \mathbb{Z}_q^{*2}$, no matter $Q = Q'$ ($m = m'$) or not.

Definition 4 (Encrypt+SoK). Let $\mathfrak{E} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let $R((Q, m; r, v), (\mathsf{C}, pk)) := (\mathsf{C} = \mathcal{E}(pk, Q; r, v), Q = h^m)$ be the relation that C is encryption of Q with randomness r and v with relation L_{valid}. Let $\mathfrak{P} = (\mathcal{P}, \mathcal{V})$ be a non-interactive SoK for this relation.

The Encrypt+SoK transformation $\mathfrak{E}_{\mathfrak{P}}$ is the following encryption scheme.

- $\mathcal{G}'(1^\lambda)$: Run $\mathcal{G}(^\lambda)$.
- $\mathcal{E}'(pk, m)$: Choose $r, v \xleftarrow{\$} \mathbb{Z}_p^2$ and compute a ciphertext $\mathsf{C} = \mathcal{E}(pk, Q; r, v)$. Create a proof $\pi \leftarrow \mathsf{sFS}.\Sigma_{valid}(pk, \mathsf{C}, Q, r, v, m)$. The ciphertext is the pair (C, π).
- $\mathcal{D}'(sk, \mathsf{C}, \pi)$: First run $\mathsf{Verify}(pk, \mathsf{C}, \pi)$. If this fails, output $\perp$ and halt. Otherwise, return $\mathcal{D}(sk, \mathsf{C})$.

One more thing left is to explore the capability of $\mathfrak{E}_{\mathfrak{p}}$ to resist active adversary attacks, so that it can be used as traceability evidence in transactions related to anonymous authentications. We reviewed three definitions of non-malleable cryptographic systems [38] and followed the original definition of "simulation-based game" [39], because "indistinguishability-based game" is not applicable here due to the one-wayness of our scheme. It is still a problem to prove the non-malleability of a OW-CPA with *simulation-sound extractable* (SSE) non-interactive SoK for the encryption relation. Intuitively speaking, the non-malleability of sFS ensures that it is difficult for adversaries to generate other $\mathfrak{E}_{\mathfrak{P}}$ ciphertexts, corresponding to the same plaintext, based on an existing ciphertext. Thus we give a conjecture below.

Conjecture 1. OW-CPA + **SSE-SoK** gives non-malleable encryption.

The decryption key allows traceability, and the non-malleability of the encryption guarantees non-frameability or exculpability, as one cannot maliciously accuse a user by generating fake encryptions.

4. Joint Signature of Knowledge for TEET

In this section, we present a signature of knowledge scheme for TEET that the prover consists of two parts: hardware and device.

The main idea is as follows. Since the secret value hid is only stored in the hardware, the device (only has $Q = h_1^{\mathsf{hid}}$) must cooperate with the hardware to produce a valid zero-knowledge proof for relation L_{valid}. Based on this observation, we modify Σ_{valid} by splitting and distributing the functionality of P to the hardware and device, as well as using the strong Fiat–Shamir heuristic [30] to get a non-interactive zero-knowledge proof with non-malleability. To reduce the computational burden on the hardware and protect the privacy of hid and Q, we directly use $Q = h_1^{\mathsf{hid}}$ as the input message of $\mathsf{TEET}.\mathcal{E}$. The device can independently generate TEET ciphertext and cooperate with the hardware to generate a valid zero-knowledge proof.

The new protocol is shown in Figure 3. The device chooses random numbers r and v, and calculates $\mathsf{C} = (X, Y, U, K)$ (TEET encryption of $Q = h_1^{\mathsf{hid}}$) for initiation. To prove that the ciphertext was constructed correctly, the device calculates D and sends pk, K, D, r to the hardware. Then the hardware calculates A, B, C, and the challenge e accordingly. To prevent the illegal query from the adversary, the hardware also additionally selects $n'_{\mathcal{H}}$ and generates r' with its PRF key K to mask its output. Finally, the device

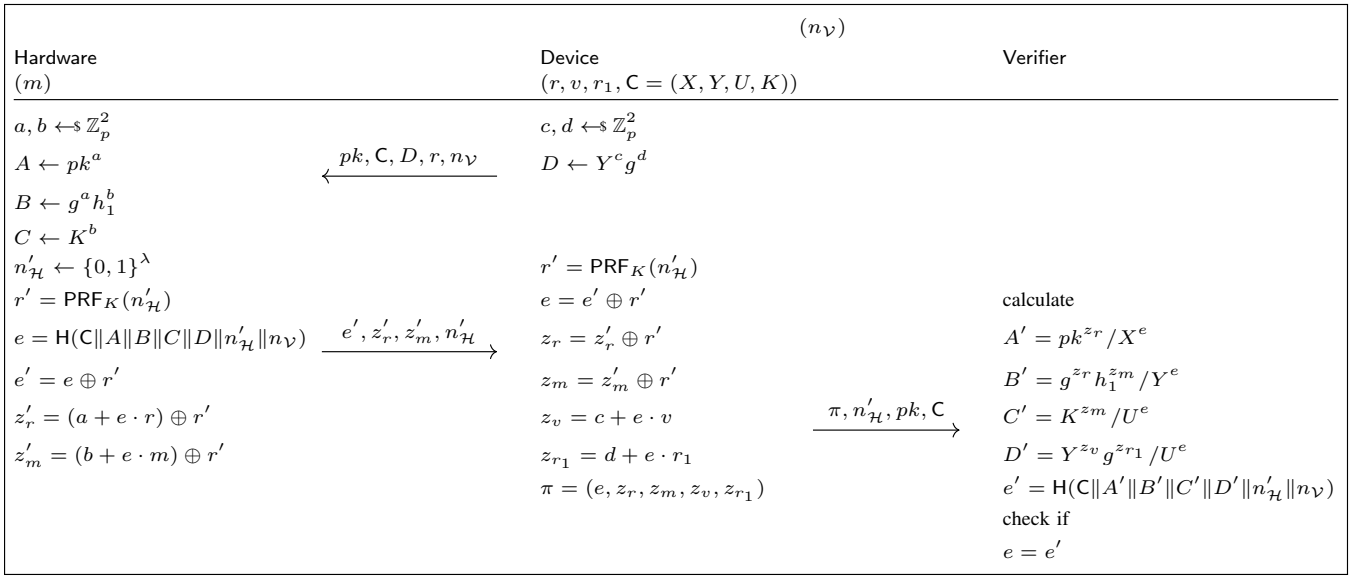

Figure 3: Cooperative version of Σ_{valid}.

generates the full proof and sends it together with pk, $n'_{\mathcal{H}}$ and the ciphertext (X, Y, U, K) to the verifier.

5. A Construction for Secure Traceable hbAC Under cbAC Setting

5.1. System Model

As shown in Figure 2, ThbAC involves five entities: users, issuers, service providers, regulators, and blockchain.

1) **User:** Users are entities (with device $\mathcal{D}$ and hardware $\mathcal{H}$) who obtain identities along with attributes from the issuer and authenticate themselves to the service providers.
2) **Issuer:** Denoted as $\mathcal{I}$. By interacting with the user, the issuer finally signs the attribute credential to the user.
3) **Service Provider:** Denoted as $\mathcal{SP}$. Before providing services, $\mathcal{SP}$ will interact with $\mathcal{D}$ and $\mathcal{H}$ to determine whether the user possesses a valid credential that fulfills the required attributes.
4) **Regulator:** Denoted as $\mathcal{R}$. Regulators are trusted parties involved in cases when law enforcement or other authorized entities require the ability to extract the real identity of a malicious user.
5) **Blockchain:** Here, we model the blockchain as a publicly accessible, globally consistent, and tamper-resistant bulletin board. All the public parameters, such as the bilinear group, public key of $\mathcal{I}$ or $\mathcal{AU}$, etc. We omit the statements related to blockchain operations for simplicity.

5.2. Syntax and Security Model

We follow the syntax in [4] and let a credential system $\Psi = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{RequestCred}, \mathsf{IssueCred}, \mathsf{ShowCred}, \mathsf{VerifyCred}, \mathsf{Test}, \mathsf{Trace}\}$ be a traceable hardware-binding anonymous credential system.

1) $\mathsf{Setup}(1^\lambda)$: Input a security parameter 1^λ, it outputs common system parameters $para$, which is a implicit input to other algorithms.
2) $\mathsf{KeyGen}(1^\lambda)$: Input a security parameter 1^λ, it invokes 3 sub-algorithms $\mathsf{HDKGen}, \mathsf{IKGen}, \mathsf{RKGen}$ to generate the corresponding key pairs for hardware/device, issuer, and regulator.
   a) $\mathsf{HDKGen}(1^\lambda)$: On input security parameter 1^λ, the hardware will receive a secret unique identifier hid and a PRF key K, and the device will receive K and a public identifier corresponds to hid. Then the regulator stores the user's information and the public identifier in a list.
   b) $\mathsf{IKGen}(1^\lambda)$: On input security parameter 1^λ, the issuer will receive a key pair $(\mathsf{isk}, \mathsf{ipk})$.
   c) $\mathsf{RKGen}(1^\lambda)$: On input security parameter 1^λ, the regulator will receive a key pair $(\mathsf{rsk}, \mathsf{rpk})$.
3) $\mathsf{RequestCred}(\mathbf{a}_D, \mathsf{hid}) \leftrightarrow \mathsf{IssueCred}(\mathbf{a}_I, \mathsf{isk})$: It is an interactive protocol between hardware/device and the service provider. On input the device attribute set $\mathbf{a}_D = \{(i, m_i) : i \in D\}$ and relevant proofs. If passed verification, then the issuer will sign a credential cred with the attributes $\mathbf{a}_D$ and $\mathbf{a}_I = \{(i, m_i) : i \in I\}$ for a hardware/device pair with secret identifier hid.
4) $\mathsf{ShowCred}(\mathbf{a}_M, \mathsf{cred}, \mathsf{hid}) \leftrightarrow \mathsf{VerifyCred}(\mathbf{a}_M, \mathsf{cp}, \mathsf{rt})$: It is an interactive protocol between hardware/device and the service provider. On input the required attribute set $\mathbf{a}_M$, credential cred, and hardware secret identifier hid, the hardware and device cooperate to generate credential proof cp and regulatory text (with proof) rt. The service provider outputs either valid(1) or invalid(0).
5) $\mathsf{Test}(\mathsf{rt}_0, \mathsf{rt}_1)$: Input two regulatory texts rt_0 and rt_1, this algorithm outputs either equal(1) or unequal(0).

6) $\mathsf{Trace}(\mathsf{rt})$: on input a regulatory text rt, this algorithm outputs a public identifier of a hardware. This achieves type (a) tracing in Figure 1.
7) $\mathsf{TraceR}(\mathsf{rsk}, \mathsf{rt}, \mathcal{L}_{\mathsf{rsg}}) \leftrightarrow \mathsf{TraceSP}(\mathcal{L}_{\mathsf{mt}}, \mathcal{L}_{\mathsf{R}})$: It is an interactive protocol between the regulator and the service providers. On input a valid regulatory text (and proof) rt, a set of round-specific generators $\mathcal{L}_{\mathsf{rsg}}$, the regulator generates a corresponding set of matching text $\mathcal{L}_{\mathsf{mt}}$ and sends them to the corresponding service providers. This algorithm outputs all authentication records from $\mathcal{L}_{\mathsf{R}}$ that corresponds to rt. This achieves type (c) tracing in Figure 1.

5.3. Design Goals

We present the design goals of our traceable hbAC:

1) **Unforgeability:** A $\mathcal{PPT}$ adversary $\mathcal{A}$ without the legal credential on requested attributes cannot forge a credential to pass the verification (see the $\mathsf{ShowCred} \leftrightarrow \mathsf{VerifyCred}$ phase in Figure 4).
2) **Blindness:** $\mathcal{I}$ learns nothing about the user's private attributes $\mathbf{a}_D$ except that these attributes satisfy the required statement C (see the $\mathsf{RequestCred} \leftrightarrow \mathsf{IssueCred}$ phase in Figure 4).
3) **Privacy Preservation:** $\mathcal{SP}$ learns nothing about the user's attributes except those that were requested (see the $\mathsf{ShowCred}$ phase mentioned below).
4) **Hardware-bidning:** This property means the credential of $\mathcal{D}$ is bound to $\mathcal{H}$ and cannot be delegated to other devices or be used without the hardware.
5) **Round Isolation:** This property means the equality test of regulatory texts (or matching text) is not feasible if they were generated within different rsg.

5.4. Generic Construction

Here we present the generic construction of our Traceable hbAC system with ℓ attributes. The main components are the card-based anonymous credential system proposed in [4] and the TEET encryption we proposed in Section 3.

As the hardware possesses the long-term secret key for all credentials, the device must cooperate to generate cp and rt. When generating cp, $\mathcal{H}$ does not communicate directly with $\mathcal{D}$ with $\mathcal{SP}$ acts as an intermediary. In contrast, the latter is generated through direct cooperation between $\mathcal{D}$ and $\mathcal{H}$. So we use a little trick here: reuse the random value $r = \mathsf{PRF}_K(n_{\mathcal{H}} \| n_{\mathcal{D}})$, which was used by $\mathcal{H}$ when generating the credential proof, to produce rt. Then present a consistency proof of $\{r, \mathsf{hid} : Y = g^r h_1^{\mathsf{hid}}, \bar{B} = \bar{g}^r \bar{h}_1^{\mathsf{hid}}\}$. By doing so, we produce an intrinsic link between cp and rt. Furthermore, since the BBS+ signature and TEET encryption rely on different types of bilinear pairs, the consistency proof also demonstrates the same secret value hid was contained both in cp and rt. This is enough to make $\mathcal{SP}$ ensure that the real identity of the anonymous user can eventually be traced. Then we give a detailed construction of hbAC in Figure 4.

5.5. Security Analysis

The construction of hbAC in Figure 4 achieves our design goals including unforgeability, blindness, privacy-preserving, hardware-binding, and round-isolating. We reduce these properties to the security of BBS+ signature and zero-knowledge proof.

Lemma 2. Our construction is unforgeable if BBS+ signature is secure under the q-SDH assumption (see Section 2.2) and the hardware is tamper-resistant.

Sketch: When $\mathcal{PPT}$ adversary $\mathcal{A}$ tries to forge a valid credential, there are two cases: 1) it forges a BBS+ signature of the issuer. Doing so contradicts the q-SDH assumption; 2) it modifies the hid in the hardware and uses the corresponding BBS+ signature. However, this is contradictory to the tamper-resistant property of a hardware. □

Lemma 3. Our construction is blind if π_2 is a signature of knowledge protocol satisfying zero-knowledge.

Sketch: In the $\mathsf{RequestCred} \leftrightarrow \mathsf{IssueCred}$ phase, the user sends the signature of knowledge $\pi_2 = \mathsf{SoK}\{(s', \{m_i\}_{i\in H}) : C = \bar{g}^{s'} \prod_{i\in H} \bar{h}_{i+1}^{m_i}\}(n_{\mathcal{I}})$ of his private attributes $\{m_i\}_{i\in H}$. The zero-knowledge property of π_2 ensures that commitment does not reveal information about these attributes. □

Lemma 4. Our construction is privacy-preserving if π_4 is a signature of knowledge protocol satisfying zero-knowledge.

Sketch: In $\mathsf{ShowCred} \leftrightarrow \mathsf{VerifyCred}$ phase, the user sends the signature of knowledge $\pi_4 \leftarrow \mathsf{SoK}\{(e, \overline{s}, r_2, r_3, \{m_i\}_{i\in H}) : A'^{-e} \bar{g}^{r_2} = \bar{A}/d \wedge d^{-r_3} \bar{g}^{s'} C = \bar{g}_1^{-1} \bar{B}^{-1} \prod_{i\in M} \bar{h}_i^{-m_i}\}(n_{\mathcal{SP}})$ to the service provider. This ensures that all unused attributes are protected. Besides, the construction of commitment $C = \bar{g}^{\overline{r}} \prod_{i\in H} \bar{h}_i^{m_i'}$ satisfies perfect hiding. So the unused attributes of the user cannot be inferred by $\mathcal{PPT}$ adversary. □

Lemma 5. Our construction is hardware-binding if π_3 and π_4 are non-interactive SoK protocols satisfying soundness and the hardware is tamper-resistant.

Sketch: In $\mathsf{ShowCred} \leftrightarrow \mathsf{VerifyCred}$ phase, $\mathcal{SP}$ execute the hardware matching first. Then the user sends the proof $\pi_3 = \mathsf{SoK}\{(\mathsf{hid}, r) : \bar{B} = \bar{h}_1^{\mathsf{hid}} \bar{g}^r\}(n_{\mathcal{SP}})$ and $\pi_4 \leftarrow \mathsf{SoK}\{(e, \overline{s}, r_2, r_3, \{m_i\}_{i\in H}) : A'^{-e} \bar{g}^{r_2} = \bar{A}/d \wedge d^{-r_3} \bar{g}^{s'} C = \bar{g}_1^{-1} \bar{B}^{-1} \prod_{i\in M} \bar{h}_i^{-m_i}\}(n_{\mathcal{SP}})$ to the service provider. As the SoK proofs are sound, the device proves that he is indeed the owner of the hardware that can provide the missing link in the verification chain. □

Lemma 6. Our construction is round-isolating if the orders of round-specific generators are unknown.

Sketch: The proof is straightforward as we already proved the computational soundness-DR of TEET in Theorem 2. For a matching text (Q^v, h_i^v) of a device with identifier Q with round h_i and a regulatory text $\mathsf{rt} = (\cdot, Q'^{v'}, h^{v'})$. The equality of $e(Q^v, h^{v'})$ and $e(Q'^{v'}, h_i^v)$ can't be tested if $\theta = \log_h h_i$ is unknown. □

$\mathsf{Setup}(1^\lambda)$:
$pp_1 = (p, \mathbb{G}_1, g, \mathbb{G}_T, e) \leftarrow \mathsf{BGGen_1}(1^\lambda)$; $h \leftarrow\!\$\ \mathbb{G}_1$
$pp_3 = (p, \dot{\mathbb{G}}_1, \dot{g}, \dot{\mathbb{G}}_2, \tilde{g}, \dot{\mathbb{G}}_T, \dot{e}) \leftarrow \mathsf{BGGen_3}(1^\lambda)$; $(\bar{h}_0, \ldots, \bar{h}_\ell) \leftarrow\!\$\ \dot{\mathbb{G}}_1^{\ell+1}$
return $pp = (pp_1, h, pp_3, \bar{h}_0, \ldots, \bar{h}_\ell)$

$\mathsf{KeyGen}(1^\lambda)$:
$\mathsf{HDKGen}(1^\lambda)$:
A trusted hardware issuer $\mathcal{F}_{\mathsf{cardAuth}}$ grants the hardware a secret identifier hid and a PRF key K, and the device $Q = g^{\mathsf{hid}}, \bar{Q} = \bar{h}_1^{\mathsf{hid}}$ and key K.
$\mathsf{IKGen}(1^\lambda)$:
$x \leftarrow\!\$\ \mathbb{Z}_p$; $w = \tilde{g}^x$; return $\mathsf{isk} = x$, $\mathsf{ipk} = w$
$\mathsf{RKGen}(1^\lambda)$:
$sk \leftarrow\!\$\ \mathbb{Z}_p$; $pk = g^{sk}$; return $\mathsf{rsk} = sk$, $\mathsf{rpk} = pk$

$\mathsf{RequestCred}(\mathbf{a}_D, \mathsf{hid}) \leftrightarrow \mathsf{IssueCred}(\mathbf{a}_I, \mathsf{isk})$: The outline of this process is shown in Figure 5.
Step (1):
$n_\mathcal{I} \leftarrow \{0,1\}^\lambda$
$\mathcal{I} \rightarrow \mathcal{H}$
Step (2):
$n_\mathcal{H} \leftarrow \{0,1\}^\lambda$; $r = \mathsf{PRF}_K(n_\mathcal{H})$; $\bar{B} = \bar{h}_1^{\mathsf{hid}} \bar{g}^r$
$\pi_1 = \mathsf{SoK}\{(\mathsf{hid}, r) : \bar{B} = \bar{h}_1^{\mathsf{hid}} \bar{g}^r\}(n_\mathcal{I})$.
$\mathcal{H} \rightarrow \mathcal{I}$
Step (3):
If π_1 is not valid; return $\perp$.
Else; $\mathcal{I} \rightarrow \mathcal{D}$
Step (4):
$s' \leftarrow \mathbb{Z}_p$; $C = \bar{g}^{s'} \prod_{i \in H} \bar{h}_{i+1}^{m_i}$; $r = \mathsf{PRF}_F(n_\mathcal{H})$
$\pi_2 = \mathsf{SoK}\{(s', \{m_i\}_{i \in H}) : C = \bar{g}^{s'} \prod_{i \in H} \bar{h}_{i+1}^{m_i}\}(n_\mathcal{I})$.
$\mathcal{D} \rightarrow \mathcal{I}$
Step (5):
If π_2 is not valid; return $\perp$.
Else; $e \leftarrow \mathbb{Z}_p \backslash \{x\}$, $\tilde{s} \leftarrow \mathbb{Z}_p$; $A = (\bar{g} \cdot \bar{g}^s \cdot \bar{B} \cdot C \cdot \prod_{i \in I} \bar{h}_{i+1}^{m_i})^{1/(e+x)}$
$\mathcal{I} \rightarrow \mathcal{D}$
$\mathbf{m} = \mathbf{a}_H \oplus \mathbf{a}_I = (m_2, \ldots, m_\ell)$
If $\dot{e}(A, w\tilde{g}^e) \neq \dot{e}(\bar{g} \cdot \bar{g}^{\tilde{s}+s'+r} \cdot \bar{Q} \cdot \prod_{i=2}^{\ell} \bar{h}_{i+1}^{m_i}, \tilde{g})$; return $\perp$
Else; $\mathcal{L}_\mathcal{C} = \mathcal{L}_\mathcal{C} \cup \mathsf{cred} = (\mathcal{C}, Q, \bar{Q}, K, \sigma = (A, e, \tilde{s} + s' + r), \mathbf{m})$

$\mathsf{Test}(\mathsf{rt}_0, \mathsf{rt}_1)$:
Return $\mathsf{TEET.Test}(\mathsf{rt}_0, \mathsf{rt}_1)$

$\mathsf{ShowCred}(\mathbf{a}_M, \mathsf{cred}, \mathsf{hid}) \leftrightarrow \mathsf{VerifyCred}(\mathbf{a}_M, \mathsf{cp}, \mathsf{rt})$: The outline of this process is shown in Figure 6.
Step (1):
$n_\mathcal{D} \leftarrow \{0,1\}^\lambda$
$\mathcal{D} \rightarrow \mathcal{SP}$
Step (2):
$n_\mathcal{SP} \leftarrow \{0,1\}^\lambda$
$\mathcal{SP} \rightarrow \mathcal{H}$
Step (3):
$n_\mathcal{H} \leftarrow \{0,1\}^\lambda$; $n = n_\mathcal{H} \| n_\mathcal{D}$, $r = \mathsf{PRF}_K(n)$; $\bar{B} = \bar{h}_1^{\mathsf{hid}} \bar{g}^r$
$\pi_3 = \mathsf{SoK}\{(\mathsf{hid}, r) : \bar{B} = \bar{h}_1^{\mathsf{hid}} \bar{g}^r\}(n_\mathcal{SP})$
$\mathcal{H} \rightarrow \mathcal{SP}$
Step (4):
If π_3 is not valid; return $\perp$
Else; $\mathcal{SP} \rightarrow \mathcal{D}$
Step (5-7):
$n = n_\mathcal{H} \| n_\mathcal{D}$; $\mathbf{a}_M = \{(i, m_i) : i \in M\}$; Find $\mathsf{cred} = (\mathcal{C}, Q, \bar{Q}, K, \sigma, \mathbf{m}) \in \mathcal{L}_\mathcal{C}$: $r = \mathsf{PRF}_K(n)$, $\bar{B} = \bar{Q} \cdot \bar{g}^r$ and $\boldsymbol{m}[i] = m_i$ for all $i \in M$. Returns $\perp$ if no record is found.
Else; $\mathbf{a}_M = \{(i, m'_{i+1}) : i \in M\}$; $\sigma = (A, e, s)$
$r = \mathsf{PRF}_K(n)$; $v \leftarrow\!\$\ \mathbb{Z}_p$; $Y = g^r h_1^{\mathsf{hid}}$; $X = pk^r$; $U = Q^v$
$K = h_1^v$; $\mathsf{rt} = (X, Y, U, K)$
$\pi_5 \leftarrow \mathsf{SoK}\{(r, \mathsf{hid}, v, r_1 = -rv) : X = pk^r \wedge Y = g^r h_1^{\mathsf{hid}} \wedge U = K^{\mathsf{hid}} \wedge U = Y^v g^{r_1} \wedge \bar{B} = \bar{g}^r \bar{h}_1^{\mathsf{hid}}\}(n_\mathcal{SP})$ ($\mathcal{D}$ cooperate with $\mathcal{H}$ as shown in Figure 3)
$\mathbf{a}_H$ as the hiding attributes ($H = \{2, \ldots, \ell\} \backslash M$)
$r_1 \leftarrow \mathbb{Z}_p^*$; $r_2, \overline{r} \leftarrow \mathbb{Z}_p^2$; $r_3 = r_1^{-1}$; $s' = s - r_2 r_3 - r$
$\overline{s} = s' - \overline{r}$; $A' = A^{r_1}$; $b = \bar{g}_1 \bar{g}^s Q \prod_{i=2}^{\ell} \bar{h}_i^{m'_i}$
$\bar{A} = A'^{-e} b^{r_1}$; $d = b^{r_1} \bar{g}^{-r_2}$; $C = \bar{g}^{\overline{r}} \prod_{i \in H} \bar{h}_i^{m'_i}$
$\pi_4 \leftarrow \mathsf{SoK}\{(e, \overline{s}, r_2, r_3, \{m'_i\}_{i \in H}) : A'^{-e} \bar{g}^{r_2} = \bar{A}/d \wedge d^{-r_3} \bar{g}^{s'} C = \bar{g}_1^{-1} \bar{B}^{-1} \prod_{i \in M} \bar{h}_i^{-m'_i}\}(n_\mathcal{SP})$.
$\mathcal{D} \rightarrow \mathcal{SP}$
If π_4, π_5 are not valid or $\dot{e}(\bar{A}, w) \neq \dot{e}(A', \tilde{g})$; return $\perp$
Else; $\mathcal{L}_\mathsf{R} = \mathcal{L}_\mathsf{R} \cup (A', \bar{A}, d, \pi_4, \mathsf{rt}, n'_\mathcal{C}, pk, \pi_5)$

$\mathsf{Trace}(\mathsf{rsk}, \mathsf{rt})$:
Return $\mathsf{TEET}.\mathcal{D}(\mathsf{rsk}, \mathsf{rt})$

$\mathsf{TraceR}(\mathsf{rsk}, \mathsf{rt}, \mathcal{L}_{\mathsf{rsg}}) \leftrightarrow \mathsf{TraceSP}(\mathcal{L}_{\mathsf{mt}}, \mathcal{L}_\mathsf{R})$:
$Q = \mathsf{Trace}(\mathsf{rsk}, \mathsf{rt})$; $\mathcal{L}_{\mathsf{mt}} = \{(Q^v, h_i^v), h_i \in \mathcal{L}_{\mathsf{rsg}}, v \leftarrow\!\$\ \mathbb{Z}_p\}_{i \in [\mathcal{L}_{\mathsf{rsg}}]}$
$\mathcal{R} \rightarrow \mathcal{SP}$
Return $\{\mathsf{R}_{i,j} | \mathsf{Test}(\mathsf{rt}, (\cdot, \mathsf{mt}_j)) = 1, \mathsf{R}_{i,j} = (\cdot, \mathsf{rt}), \mathsf{mt}_j = (Q^v, h_j^v)\}_{i \in \mathcal{L}_\mathsf{R} \wedge j \in \mathcal{L}_{\mathsf{mt}}}$

Figure 4: Our generic construction of hbAC.

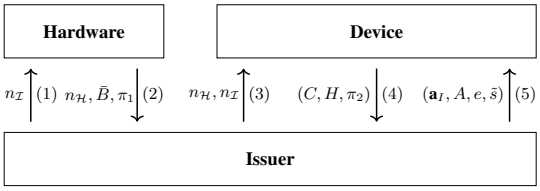

Figure 5: Outline of credential issuance between Hardware, Device, and Issuer. Skeleton borrowed from [4].

6. Performance Analysis

The execution time of each operation is given in Table 2. We analyze the efficiency of our construction by counting different operations, i.e., T_{cmp}, T_{mul}, T_{add}, and T_{hash}. We also give the computation cost analysis of our construction in Table 3. In $\mathsf{RequestCred} \leftrightarrow \mathsf{IsseuCred}$ phase h and i refer to the number of attributes held by H and I. In $\mathsf{ShowCred} \leftrightarrow \mathsf{VerifyCred}$ phase h and m refer to the unused number of attributes held by H and the number of attributes required by I. We ignore the computation cost related to $\mathcal{F}_{\mathsf{cardAuth}}$, so the total computation cost in HDKGen phase is 0. We use gray text to mark the additional cost introduced by our

TABLE 2: Execution time of each operation

Notions	Description
T_{cmp}	Compare two pairings
T_{mul}	Multiplication operation in $\mathbb{G}$
T_{add}	Add operation in $\mathbb{G}$
T_{hash}	Hash operation
$\lvert\mathbb{G}_1\rvert$	Bit length of an element in $\mathbb{G}_1$
$\lvert\mathbb{G}_T\rvert$	Bit length of an element in $\mathbb{G}_T$
$\lvert\dot{\mathbb{G}}_1\rvert$	Bit length of an element in $\dot{\mathbb{G}}_1$
$\lvert\dot{\mathbb{G}}_2\rvert$	Bit length of an element in $\dot{\mathbb{G}}_2$
$\lvert\dot{\mathbb{G}}_T\rvert$	Bit length of an element in $\dot{\mathbb{G}}_T$
$\lvert\mathbb{Z}_p\rvert$	Bit length of an element in $\mathbb{Z}_p$

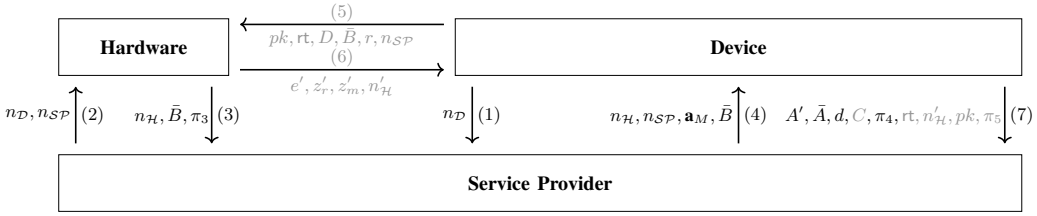

Figure 6: Outline of credential presentation between Hardware, Device, and Service Provider. Skeleton borrowed from [4]. The gray text denotes the messages we used to generate TEET ciphertext and the corresponding noninteractive zero-knowledge proof.

TABLE 3: Computation Cost of Our Construction

Algorithms		Computation cost
KeyGen	$\mathcal{H}\&\mathcal{D}$	0
	$\mathcal{I}$	1 T_{mul}
	$\mathcal{R}$	1 T_{mul}
RequestCred	$\mathcal{H}$	4 T_{mul} + 2 T_{add} + 1 T_{hash}
	$\mathcal{D}$	2 T_{cmp} + (3 + 2 $\lvert\mathbf{a}_H\rvert$ + ℓ) · T_{mul} + (2 + 2 $\lvert\mathbf{a}_H\rvert$ + ℓ) · T_{add} + 1 T_{hash}
IssueCred	$\mathcal{I}$	(6 + ℓ) · T_{mul} + (7 + ℓ) · T_{add} + 2 T_{hash}
ShowCred	$\mathcal{H}$	(4 T_{mul} + 2 T_{add} + 1 T_{hash}) + (4 T_{mul} + 1 T_{add} + 1 T_{hash})
	$\mathcal{D}$	(10 + 7 + $\lvert\mathbf{a}_H\rvert$ + ℓ) · T_{mul} + (4 + 2 + $\lvert\mathbf{a}_H\rvert$ + ℓ) · T_{add} + 1 T_{hash}
VerifyCred	$\mathcal{SP}$	1 T_{cmp} + (10 + 13 + $\lvert\mathbf{a}_M\rvert$) · T_{mul} + (14 + 8 + $\lvert\mathbf{a}_M\rvert$) · T_{add} + (1 + 1) · T_{hash}
Test		1 T_{cmp}
Trace	$\mathcal{R}$	1 T_{mul} + 1 T_{add} + 1 T_{cmp}
TraceR	$\mathcal{R}$	1 T_{mul} + (1 + $\lvert\mathcal{L}_{\mathsf{rsg}}\rvert$) · T_{add} + 1 T_{cmp}
TraceSP	$\mathcal{SP}$	$\lvert\mathcal{L}_{\mathsf{R}}\rvert$ · T_{cmp}

TABLE 4: Communication Cost of Our Construction

Algorithms		Communication cost
RequestCred ↔ IsseuCred	$\mathcal{I} \to \mathcal{H}$	1 $\lvert\lambda\rvert$
	$\mathcal{H} \to \mathcal{I}$	1 $\lvert\lambda\rvert$ + 3 $\lvert\mathbb{Z}_p\rvert$ + 1 $\lvert\dot{\mathbb{G}}_1\rvert$
	$\mathcal{I} \to \mathcal{D}$	2 $\lvert\lambda\rvert$
	$\mathcal{D} \to \mathcal{I}$	(2 + $\lvert\mathbf{a}_H\rvert$) · $\lvert\mathbb{Z}_p\rvert$ + 1 $\lvert\dot{\mathbb{G}}_1\rvert$ + $\lvert\mathbf{a}_H\rvert$ · $\lvert tag\rvert$
	$\mathcal{I} \to \mathcal{D}$	(2 + $\lvert\mathbf{a}_I\rvert$) · $\lvert\mathbb{Z}_p\rvert$ + 1 $\lvert\dot{\mathbb{G}}_1\rvert$
ShowCred ↔ VerifyCred	$\mathcal{D} \to \mathcal{SP}$	1 $\lvert\lambda\rvert$
	$\mathcal{SP} \to \mathcal{H}$	2 $\lvert\lambda\rvert$
	$\mathcal{H} \to \mathcal{SP}$	1 $\lvert\lambda\rvert$ + 3 $\lvert\mathbb{Z}_p\rvert$ + 1 $\lvert\dot{\mathbb{G}}_1\rvert$
	$\mathcal{SP} \to \mathcal{D}$	(2 + $\mathbf{a}_M$) · $\lvert\mathbb{Z}_p\rvert$ + 1 $\lvert\dot{\mathbb{G}}_1\rvert$
	$\mathcal{D} \to \mathcal{H}$	1 $\lvert\lambda\rvert$ + 1 $\lvert\mathbb{Z}_p\rvert$ + 1 $\lvert\dot{\mathbb{G}}_1\rvert$ + 6 $\lvert\mathbb{G}_1\rvert$
	$\mathcal{H} \to \mathcal{D}$	1 $\lvert\lambda\rvert$ + 3 $\lvert\mathbb{Z}_p\rvert$
	$\mathcal{D} \to \mathcal{SP}$	(5 + 6) · $\lvert\mathbb{Z}_p\rvert$ + 3 $\lvert\dot{\mathbb{G}}_1\rvert$ + 5·$\lvert\mathbb{G}_1\rvert$

scheme in Table 3 and 4 when comparing with [4].

For the communication cost of our construction, we show it in Table 4. The communication overhead is mainly concentrated on the phases related to the transfer of attributes, while the costs for other phases are fixed.

7. Conclusion

In this paper, we introduce the traceable anonymous credential with hardware-binding (hbAC). **traceable** means the true identity of the credential owner can be traced through the records although the authentication process is anonymous. **hb** means proving possession of credentials (as well as attributes) requires the participant of a specific hardware. Our solution involves the design of a new public key encryption scheme with equality test and verifiability which is independent of interest. With the unique and unknown order round-specific generator for each authentication round, we both achieve user privacy and secure outsourcing of identity tracing tasks. Besides, we also propose a non-interactive signature of knowledge scheme to implement collaborative knowledge proof between the hardware and device.

Open Problems and Future Work. We still have some problems: 1) the non-malleability of an OW-CPA encryption scheme with **SSE** signature of knowledge is not formally proved. We only give a conjecture based on existing works. 2) Cooperative SNARKs between normal devices and hardware with limited capabilities is another attractive technology, which can help us turn the hash based encryption schemes into verifiable on and combine with hbAC. While the merit of hardware-binding in AC systems is obvious, as they can help preventing unauthorized credential sharing or even achieves human-binding, hardware also can play an important role in areas such as property protection and data sharing.

Appendix

Here are the details of the SoK we used.

1) $\pi_1 = \mathsf{SoK}\{(\mathsf{hid}, r) : \bar{h}_1^{\mathsf{hid}}\bar{g}^r = \bar{B}\}(n_{\mathcal{I}})$.
   a) $\mathcal{C}$ randomly selects $r_{\mathsf{hid}}, r_r \leftarrow\$ \mathbb{Z}_p^2$, calculates $\overline{B} = \bar{h}_1^{r_{\mathsf{hid}}}\bar{g}^{r_r}$, $e = \mathsf{H}(\bar{B}\|\overline{B}\|n_{\mathcal{I}})$, $z_{\mathsf{hid}} = r_{\mathsf{hid}} + e \cdot \mathsf{hid}$, $z_r = r_r + e \cdot r$, and sends $(e, z_{\mathsf{hid}}, z_r)$.
   b) The proof will be accepted if $e = \mathsf{H}(\bar{B}\| h_1^{z_{\mathsf{hid}}} g^{z_r}/\bar{B}^e\|n_{\mathcal{I}})$ holds.
2) $\pi_2 = \mathsf{SoK}\{(s', \{m_i\}_{i\in H}) : C = \bar{g}^{s'}\prod_{i\in H}\bar{h}_{i+1}^{m_i}\}(n_{\mathcal{I}})$.
   a) $\mathcal{H}$ randomly selects $r_{s'}, \{r_{m_i}\}_{i\in H} \leftarrow\$ \mathbb{Z}_p^{|H|+1}$, calculates $\overline{C} = \bar{g}^{r_{s'}}\prod_{i\in H}\bar{h}_{i+1}^{r_{m_i}}$, $e = \mathsf{H}(C\|\overline{C}\|n_{\mathcal{I}})$,

$z_{s'} = r_{s'} + e \cdot s'$, $\{z_{m_i} = r_{m_i} + e \cdot m_i\}_{i \in H}$, and sends $(e, z_{s'}, \{z_{m_i}\})$.

b) The the proof will be accepted if $e = \mathsf{H}(C \| \bar{g}^{z_{s'}} \prod_{i \in H} \bar{h}_{i+1}^{z_{m_i}} / \ C^e \| n_{\mathcal{I}})$ holds.

3) π_3: consistent with π_1.

4) $\pi_4 \leftarrow \mathsf{SoK}\{(e, \bar{s}, r_2, r_3, \{m_i'\}_{i \in H}) : A'^{-e} \bar{g}^{r_2} = \bar{A}/d \wedge d^{-r_3} \bar{g}^{s'} C = \bar{g}_1^{-1} \bar{B}^{-1} \prod_{i \in M} \bar{h}_i^{-m_i'}\}(n_{\mathcal{SP}})$.

a) $\mathcal{H}$ randomly selects $r_e, r_{\bar{s}}, r_{r_2}, r_{r_3} \leftarrow\$ \mathbb{Z}_p^4$, calculates $\Theta \leftarrow A'^{-r_e} \bar{g}^{r_{r_2}}$, $\Xi \leftarrow d^{-r_{r_3}} \bar{g}^{r_{\bar{s}}}$, $e = \mathsf{H}(\Theta \| \Xi \| n_{\mathcal{I}})$, $z_e = e + c \cdot e$, $z_{\bar{s}} = r_{\bar{s}} + c \cdot \bar{s}$, $z_{r_2} = r_{r_2} + c \cdot r_2$, $z_{r_3} = r_{r_3} + c \cdot r_3$, and sends $(e, z_e, z_{\bar{s}}, z_{r_2}, z_{r_3})$.

b) Accept the proof if $e = \mathsf{H}(A'^{-z_e} \bar{g}^{z_{r_2}} / (\frac{\bar{A}}{d})^c \| d^{-z_{r_3}} \bar{g}^{z_{\bar{s}}} C^e / (\bar{g}_1^{-1} \bar{B}^{-1} \prod_{i \in V} \bar{h}_i^{-m_i'})^c \| n_{\mathcal{I}})$ holds.

5) $\pi_5 \leftarrow \mathsf{SoK}\{(r, \mathsf{hid}, v, r_1 = -rv) : X = pk^r \wedge Y = g^r h_1^{\mathsf{hid}} \wedge U = K^{\mathsf{hid}} \wedge U = Y^v g^{r_1} \wedge \bar{B} = \bar{g}^r \bar{h}_1^{\mathsf{hid}}\}(n_{\mathcal{SP}})$.

a) $\mathcal{D}$ randomly selects $c, d \leftarrow\$ \mathbb{Z}_p^2$, calculates $D = Y^c g^d$. Then sends $pk, \mathsf{rt} = \{X, Y, U, K\}, D, r, n_{\mathcal{SP}}$ to $\mathcal{H}$.

b) $\mathcal{H}$ randomly selects $a, b \leftarrow\$ \mathbb{Z}_p^2$ and $n'_{\mathcal{H}} \leftarrow \{0,1\}^\lambda$, calculates $A \leftarrow pk^a$, $B \leftarrow g^a h_1^b$, $\overline{B} \leftarrow \bar{g}^a \bar{h}_1^b$, $C \leftarrow K^b$, $r' = \mathsf{PRF}_K(n'_{\mathcal{H}})$, $e = \mathsf{H}(\mathsf{rt} \| A \| B \| \overline{B} \| C \| D \| n'_{\mathcal{H}} \| n_{\mathcal{SP}})$, $e' = e \oplus r'$, $z'_r = (a + e \cdot r) \oplus r'$, $z'_{\mathsf{hid}} = (b + e \cdot \mathsf{hid}) \oplus r'$. Then returns $(e', z'_r, z'_{\mathsf{hid}}, n'_{\mathcal{H}})$ to $\mathcal{D}$.

c) $\mathcal{D}$ calculates $r' = \mathsf{PRF}_K(n'_{\mathcal{H}})$, $e = e' \oplus r'$, $z_r = z'_r \oplus r'$, $z_{\mathsf{hid}} = z'_{\mathsf{hid}} \oplus r'$, $z_v = c + e \cdot v$, $z_{r_1} = d + e \cdot r_1$. Then sends $(e, z_r, z_{\mathsf{hid}}, z_v, z_{r_1})$.

d) The proof will be accepted if $e = \mathsf{H}(\mathsf{rt} \| pk^{z_r} / X^e \| g^{z_r} h_1^{z_{\mathsf{hid}}} / Y^e \| \bar{g}^{z_r} \bar{h}_1^{z_{\mathsf{hid}}} / \overline{B}^e \| K^{z_{\mathsf{hid}}} / U^e \| Y^{z_v} g^{z_{r_1}} / U^e \| n'_{\mathcal{H}} \| n_{\mathcal{SP}})$ holds.

References

[1] J. Camenisch and T. Groß, "Efficient attributes for anonymous credentials," *ACM Transactions on Information and System Security (TISSEC)*, vol. 15, no. 1, pp. 1–30, 2012.

[2] H. Yu, C. Du, Y. Xiao, A. Keromytis, C. Wang, R. Gazda, Y. T. Hou, and W. Lou, "Aaka: An anti-tracking cellular authentication scheme leveraging anonymous credentials," in *Network and Distributed System Security Symposium (NDSS)*, 2024.

[3] L. Hanzlik and D. Slamanig, "With a little help from my friends: Constructing practical anonymous credentials," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2004–2023.

[4] J. Hesse, N. Singh, and A. Sorniotti, "How to bind anonymous credentials to humans," in *USENIX Security Symposium*, 2023.

[5] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 132–145.

[6] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic macs and keyed-verification anonymous credentials," in *Proceedings of the 2014 acm sigsac conference on computer and communications security*, 2014, pp. 1205–1216.

[7] J. Camenisch, M. Drijvers, P. Dzurenda, and J. Hajny, "Fast keyed-verification anonymous credentials on standard smart cards," in *ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34*. Springer, 2019, pp. 286–298.

[8] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup, "Anonymous credentials on a standard java card," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 600–610.

[9] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20*. Springer, 2001, pp. 93–118.

[10] M. Kohlweiss, A. Lysyanskaya, and A. Nguyen, "Privacy-preserving blueprints," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2023, pp. 594–625.

[11] S. Canard and R. Lescuyer, "Protecting privacy by sanitizing personal data: a new approach to anonymous credentials," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 381–392.

[12] K. Wang, J. Gao, Q. Wang, J. Zhang, Y. Li, Z. Guan, and Z. Chen, "Hades: Practical decentralized identity with full accountability and fine-grained sybil-resistance," in *Proceedings of the 39th Annual Computer Security Applications Conference*, 2023, pp. 216–228.

[13] K. Bemmann, J. Blömer, J. Bobolz, H. Bröcher, D. Diemert, F. Eidens, L. Eilers, J. Haltermann, J. Juhnke, B. Otour *et al.*, "Fully-featured anonymous credentials with reputation system," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–10.

[14] C. Hébant and D. Pointcheval, "Traceable constant-size multi-authority credentials," in *International Conference on Security and Cryptography for Networks*. Springer, 2022, pp. 411–434.

[15] K. Panahi, S. Robertson, Y. Acar, A. G. Bardas, T. Kohno, and L. Simko, ""but they have overlooked a few things in {Afghanistan:}" an analysis of the integration of biometric voter verification in the 2019 afghan presidential elections," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2047–2064.

[16] B. Adida, "Helios: Web-based open-audit voting." in *USENIX security symposium*, vol. 17, 2008, pp. 335–348.

[17] V. Cortier and B. Smyth, "Attacking and fixing helios: An analysis of ballot secrecy," *Journal of Computer Security*, vol. 21, no. 1, pp. 89–148, 2013.

[18] P. Chaidos, V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A non-interactive receipt-free electronic voting scheme," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1614–1625.

[19] C. Paquin and G. Zaverucha, "U-prove cryptographic specification v1. 1," *Technical Report, Microsoft Corporation*, 2011.

[20] J. Camenisch *et al.*, "Specification of the identity mixer cryptographic library," *IBM Research–Zurich*, pp. 1–48, 2010.

[21] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Annual international cryptology conference*. Springer, 2004, pp. 41–55.

[22] L. Tobias, K. Vasilis, W. Andrew, and L. Mike, "The bbs signature scheme. internet-draft draft-irtf-cfrgbbs-signatures-01," Internet Engineering Task Force, last updated: 2023-03-09. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/01/

[23] A. Connolly, P. Lafourcade, and O. Perez Kempner, "Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes," in *IACR International Conference on Public-Key Cryptography*. Springer, 2022, pp. 409–438.

[24] O. Mir, B. Bauer, S. Griffy, A. Lysyanskaya, and D. Slamanig, "Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 30–44.

[25] F. Baldimtsi and A. Lysyanskaya, "Anonymous credentials light," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1087–1098.

[26] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong, "Probabilistic public key encryption with equality test," in *Topics in Cryptology-CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*. Springer, 2010, pp. 119–131.

[27] Y. Chen, X. Ma, C. Tang, and M. H. Au, "Pgc: decentralized confidential payment system with auditability," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 591–610.

[28] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for {Zero-Knowledge} proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 519–535.

[29] A. Chiesa, R. Lehmkuhl, P. Mishra, and Y. Zhang, "Eos: Efficient private delegation of {zkSNARK} provers," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 6453–6469.

[30] D. Bernhard, O. Pereira, and B. Warinschi, "How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios," in *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*. Springer, 2012, pp. 626–643.

[31] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

[32] M. Chase and A. Lysyanskaya, "On signatures of knowledge," in *Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26*. Springer, 2006, pp. 78–96.

[33] X. Zhou, D. He, J. Ning, M. Luo, and X. Huang, "Aadec: Anonymous and auditable distributed access control for edge computing services," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 290–303, 2022.

[34] D. Boneh and X. Boyen, "Short signatures without random oracles and the sdh assumption in bilinear groups," *Journal of cryptology*, vol. 21, no. 2, pp. 149–177, 2008.

[35] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*. ACM, 2019, pp. 203–225.

[36] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient public key encryption with equality test supporting flexible authorization," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 458–470, 2014.

[37] W. Susilo, F. Guo, Z. Zhao, Y. Jiang, and C. Ge, "Secure replication-based outsourced computation using smart contracts," *IEEE Transactions on Services Computing*, 2023.

[38] M. Bellare and A. Sahai, "Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization," in *Annual International Cryptology Conference*. Springer, 1999, pp. 519–536.

[39] D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography," in *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, 1991, pp. 542–552.