**Research Article**

**Publicly Traceable Attribute-Based Anonymous Authentication and Its Application to Voting**

Peng Li, Junzuo Lai, and Yongdong Wu

College of Information Science and Technology, Jinan University, Guangzhou 510632, China

Correspondence should be addressed to Junzuo Lai; laijunzuo@gmail.com

Received 28 December 2020; Revised 10 May 2021; Accepted 26 June 2021; Published 10 August 2021

Academic Editor: Savio Sciancalepore

Copyright © 2021 Peng Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Numerous anonymous authentication schemes are designed to provide efficient authentication services while preserving privacy. Such schemes may easily neglect access control and accountability, which are two requirements that play an important role in some particular environments and applications. Prior designs of attribute-based anonymous authentication schemes did not concentrate on providing full anonymity while at the same time holding public traceability. To address this problem, we formally define and present a new primitive called *traceable attribute-based anonymous authentication* (TABAA) which achieves (i) full anonymity, i.e., both registration and authentication cannot reveal user’s privacy; (ii) reusable credential, i.e., a registered credential can be repeatedly used without being linked; (iii) access control, i.e., only when the user’s attribute satisfies the access policy can the user be involved in authentication; and (iv) public traceability, i.e., anyone, without help from the trusted third party, can trace a misbehaving user who has authenticated two messages corresponding to a common address. Then, we formally define the security requirements of TABAA, including unforgeability, anonymity, and accountability, and give a generic construction satisfying the security requirements. Furthermore, based on TABAA, we propose the first attribute-based, decentralized, fully anonymous, publicly traceable e-voting, which enables voters to engage in a number of different voting activities without repeated registration.

1. **Introduction**

Consider a conventional scenario of authentication: when a user wants to access an application service, he should be authenticated by submitting his signature to an application provider; then, the latter can verify the qualification of the former. However, it brings a problem that the provider does have the ability to know who is asking for the application service. This ordinary authentication scheme seriously impairs user’s privacy.

Anonymity seems to be an appealing notion from the user’s standpoint and is a significant form of privacy preserving. Anonymous authentication is a typical protocol that achieves the authentication process while holding the anonymity, such as anonymous credential (AC), group signature, and ring signature. In an AC scheme, only those users who have registered to an organization can authenticate their membership in the organization to any verifier anonymously among all members [1]. AC was envisioned by David Chaum [2], fully formalized by Camenisch and Lysyanskaya [3], and considered as a crucial building block in privacy-concerned identity management system. As they allow users to demonstrate ownership of credentials without disclosing any other information about themselves; when such a proof is carried out, it cannot be linked to previous uses of the same credential or any other identifying information about the user [4]. A group signature scheme and a ring signature scheme can also be considered as a kind of anonymous authentication scheme, in which any group member can sign on behalf of the group while keeping anonymous. Even though anonymity can bring better privacy protection, it gives rise to severe tensions if anonymity is abused and can also potentially be an extremely dangerous tool against public safety [5]. Most authentication schemes require users to register a certificate by submitting identity information (e.g., attributes) in plaintext without any
Protection; in other words, the issuer of a certificate always learns the certificates' plaintext attributes. They only ensure the privacy preserving in authentication, so that they cannot achieve full anonymity. Thus, we also investigate how to ensure the privacy in registration process.

Accountable anonymous authentication [6] can be viewed as a special type of anonymous authentication, as the former possesses an additional significant property of accountability. Undoubtedly user’s privacy should be protected, while at the same time an essential accountability mechanism is required to check whether privacy abuse happens, or some enforcement mechanisms can even be capable of revoking such privacy if malicious behavior is detected. There are two kinds of accountability, i.e., linkability and traceability. A lightweight accountability is to achieve the linkability of multiple authentications so as to determine whether they originate from the same user, such as linkable group signature [7] and linkable ring signature [8]. A stronger type of accountability is traceability, which aims to trace the malicious user’s identity, such as group signature [9], accountable ring signature [10], and tracing-by-linking group signature [11].

On the other hand, as a crucial role in some anonymous applications, access control constrains what a user can do, as well as who has the authorization to do. For example, in e-voting, only eligible voters have the right to vote; thus, the access policy can be designated such that the age of each voter must be up to eighteen. A generalized notion of anonymous credential is attribute-based anonymous credential (ABAC) [1], which realizes the purpose of anonymous authentication such that the verifer only knows whether an authenticating user satisfies an access control policy. Similar to ABAC, an attribute-based signature (ABS) [12] extends the identity-based signature and enables a user to sign a message with fine grained control over identifying information, such that the verifier can be convinced of the fact that the signer’s attributes satisfy the access policy while keeping ignorant of the identity of the signer [13]. Particularly, the signature can hide the attributes used to satisfy the signing policy. In addition, traceable attribute-based signature (TABS) [14] and traceable attribute-based anonymous credential (TABAC) [15] can provide a good balance between privacy and accountability, but they always need a trust party to achieve the traceability.

In summary, there is no known anonymous authentication scheme that simultaneously achieves attribute protection in the registration process, while supporting an access control and public traceability. Therefore, we propose a new primitive named TABAA to achieve this goal. TABAA is suitable for many privacy-preserving applications, such as crowdsourcing and e-voting. In crowdsourcing, a requester posts a task with rewards; then, all qualified workers who satisfy a designated access policy can join in the task and obtain a reward. On the one hand, users in crowdsourcing require strong anonymity to avoid privacy disclosure, especially when the certificate authority holds their private information that may be attacked and leaked. On the other hand, it is crucial to trace the malicious users when they misuse the strong privacy and attempt to obtain more benefits by submitting multiple times in a task. Thus, TABAA is helpful to balance privacy and accountability while providing access control. In Section 6, we apply TABAA to e-voting and present a privacy-preserving e-voting scheme.

Our main contributions include the following:

1. We introduce a new notion called traceable attribute-based anonymous authentication (TABAA) which guarantees enhanced privacy protection such that a user (i) can acquire an attribute-based anonymous credential without revealing the content of his attributes and (ii) can also anonymously authenticate messages without revealing his identity. More attractively, this kind of anonymous credential can be used repeatedly without being identified. In most cases, the authentications can provide strong anonymity and unlinkability such that no one, even the certificate authority, can tell by whom they are generated. On the other hand, anyone who has authenticated two messages sharing a common event can be detected, and the one’s identity will simultaneously be revealed without any help from the certificate authority. Meanwhile, it remains anonymous if one authenticates a message only once or multiple messages with different events. More importantly, an attribute-based access policy is embedded to prevent unqualified users from authenticating messages.

2. We formally define the security requirements of TABAA, including unforgeability, full anonymity, and accountability. We further give a generic construction and reduce the security to some intractability assumptions. We also provide a comparison between our scheme and other works on attribute-based anonymous authentication schemes in Table 1. In addition, if lattice-based anonymous credential schemes and zk-SNARK protocols are used, we can instantiate our generic scheme to obtain the first lattice-based TABAA scheme, which is secure in the quantum computing model.

3. We apply the proposed TABAA to e-voting and design the first decentralized, fully anonymous, publicly traceable e-voting based on blockchain, which fulfills the following: (i) anonymous registration, i.e., a user registers a credential without privacy leakage; (ii) reusable credential, i.e., one can attend different voting activities without multiple registrations; (iii) attribute-based access control, i.e., only the one who satisfies the access policy is allowable to vote; (iv) anonymous authentication, i.e., anyone cannot tell the relation between ballot and voter; and (v) public traceability, i.e., double voting can be detected and will lead to identity exposure. Finally, the security analysis is provided to illustrate the feasibility and usability of our e-voting scheme.
The paper is organized as follows. We present the related work and essential building blocks in Section 2 and Section 3, respectively. In Section 4, we define our traceable attribute-based anonymous authentication, and in Section 5, we present a generic construction and its security proof. Next, we elaborate our voting scheme and analyze this protocol in terms of correctness and security in Section 6. Finally, we conclude our work in Section 7.

2. Related Work

2.1. Attribute-Based Anonymous Authentication. Maji et al. [12] introduced ABS which enables messages to be signed with respect to a signing policy, and provided a generic construction together with three practical instantiations. Li et al. [18] and Ge et al. [19] proposed concrete ABS schemes to support flexible threshold predicate. To reduce the trust of the central attribute authority, the authors of [12, 18] also extended the multi-authority setting. Okamoto and Takashima [20] presented the first decentralized multi-authority attribute-based signature scheme, which does not require central authority and trusted setup. All above ABS schemes ensure the basic access control and anonymity, but they lack the important feature of accountability. Even though the authors of [27] designed the attribute-based ring signature (ABRS), the construction has neither the traceability nor the basic access control. Garman et al. [24] introduced a decentralized anonymous credential to eliminate the need for a trusted credential issuer; Tan and Groß [26] proposed an ABAC scheme that supports proofs for complex statements. The authors of [24, 26] encoded a set of attributes in a credential without revealing the real content of the attributes and also exploited the access policy while lacking the ability to trace.

2.2. Accountable Attribute-Based Anonymous Authentication. The authors of [14, 28] present a kind of TABS scheme that aims to balance the usage of attribute-based signatures, such that a malicious signature can be traced. Two trusted parties (i.e., attribute issuing authority and private key generator) in [28] can work together to realize the traceability; the study in [14] only needs the attribute authority to trace a signer. However, neither of them achieves the attribute protection and public traceability. In [22, 29], the authors focus on designing the decentralized traceable attribute-based signature (DTABS) where attribute management is distributed among multiple authorities instead of relying on a single central authority. Yang et al. [30] constructed a revocable ABRBS scheme with constant-size signature, but their construction only achieves anonymity and revocability. Ali and Amberker [23] constructed an attribute-based group signature (ABGS) without random oracles, which is a kind of group signature scheme where a signature is generated by the group members possessing certain attributes. Their scheme provides attribute anonymity and user anonymity, and there exists a group manager to reveal the signer’s identity. Kuchta et al. [31] expanded the ABS to design the generic framework of an ABGS, in which the group manager and the key issuer are separated to avoid signature forgery or collusion with other members. Although the work in [31] also ensures user anonymity and attribute anonymity, it cannot achieve the public traceability. Revocable ABS was proposed in [32], which admits general signing policies and sets an external judge to break the anonymity of a signature. In addition, based on ABS, Kaaniche and Laurent [21] proposed a new anonymous credential scheme, which is made to be suitable for supporting access control policy and allows the role of inspector to remove the anonymity of a user when needed. Although the study in [21] enables user’s anonymous authentication with a verifier, it cannot implement the attribute protection and the public traceability. Hébant and Pointcheval proposed [15] a TABAC scheme that allow traceability in case of abuse of credentials, but the tracing authority is needed, and they also did not consider access policy. Similarly, Hajny et al. [16] presented an ABAC scheme supporting practical revocation of users, and Derler et al. [17] proposed a revocable attribute-based anonymous credential (RABAC) scheme, but they cannot achieve the public traceability and hold the functionality of access control.

3. Building Blocks

3.1. Notations. Throughout this paper, we denote by $\lambda$ the security parameter and by $1^\lambda$ its unary representation. A probabilistic polynomial time algorithm (PPT) runs in time polynomial in the security parameter $\lambda$. A function $f(\lambda)$ is negligible if for every $c > 0$ there exists a $\lambda_c$ such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$. An event depending on $\lambda$ occurs with overwhelming probability when its probability is at least $1 - \text{negl}(\lambda)$ for a negligible function $\text{negl}(\lambda)$. We denote by $\|\|$ the concatenation of two strings. We assume that every algorithm outputs a special symbol $\perp$ on error.

3.2. Commitment Scheme. A commitment scheme is a two-phase scheme, which enables a user to commit to a secret value, while preserving the ability of this user to open the committed value at a later phase. The standard definition of a

---

Table 1: Comparison between our scheme and related schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Attribute-protected</th>
<th>Access policy</th>
<th>Anonymity</th>
<th>Traceability</th>
<th>Public traceability</th>
</tr>
</thead>
<tbody>
<tr>
<td>[15–17]</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>[12, 18–20]</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>[14, 21–23]</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>[24–26]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Ours</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

"Attribute-protected" means that the attributes are not revealed in plaintext. Here, we use ✓ to represent a realized functionality; otherwise, we use ×.

---
noninteractive commitment scheme consists of three algorithms (ComSetup, Commit, ComVerify):

ComSetup($1^k$) is the setup algorithm that generates a public key $cpp$, given a security parameter $\lambda$.

Commit($cpp, m, r$) is the commitment algorithm that takes as input a public key $cpp$, a message $m$, and an opening $r$. It outputs a commitment $C$ on $m$.

ComVerify($cpp, m, r, C$) is the verification algorithm that takes as input a public key $cpp$, a message $m$, an opening $r$, and a commitment $C$. It outputs 1 if $C$ is a valid commitment to the message $m$, or 0 otherwise.

A commitment scheme is said to be hiding if the value committed to remains unknown before the opening phase; i.e., given $cpp$ and $C$, it is intractable to get any information about $m$. The commitment scheme is said to be binding if the only value that a commitment can be opened to is the one which is chosen in the commitment phase; i.e., given $cpp$, it is tough to figure out two different pairs $(m, r)$ and $(m', r')$ such that $\text{Commit}(cpp, m, r) = \text{Commit}(cpp, m', r')$.

3.3. Blind Signature. A blind signature scheme, first introduced by Chaum [2], is an interactive protocol between a signer and a user. The goal of blind signature schemes is to enable a user to get a signature from a signer so that the signer and a user. The signing protocol is noninteractive, in which the common input is a public key $pk$. The user’s private input is a message $m$, and the signer’s private input is the secret key $sk$. At the end of the protocol, the signer outputs either “completed” or “non-completed,” the user outputs either a signature $\sigma$ or “failed.”

SigVerify($m, \sigma, pk$) is the verification algorithm that outputs 1 if $\sigma$ is a valid signature for a message $m$, or 0 otherwise, given a message-signature pair $(m, \sigma)$ and a public key $pk$.

A blind signature scheme has to satisfy two key properties: blindness and one-more unforgeability.

3.4. Anonymous Credential. An anonymous credential, which enables users to demonstrate possession of a credential without leaking any information other than the fact that he has such a credential [3], has drawn considerable attention in recent years. In [4], such an anonymous credential scheme provides users with a credential in a confidential manner; for instance, a user should offer a set of attributes that will be signed by the issuer, and the issuer can verify that what he is signing is really the correct attribute set even if he cannot see the actual content. Specifically, a user, usually with a set of attributes, should commit these attributes to form a commitment and prove in zero knowledge that this commitment does include these attributes, and he will get a corresponding credential after running a particular blind signature protocol. When he wants to demonstrate that he really meets an access policy, he can simply show that he possesses a credential with the desired attribute.

An anonymous credential scheme introduced in [4] can be depicted as three algorithms (Setup, CreGen, Verify), where CreGen is an interactive protocol.

Setup($1^k$) is the setup algorithm that takes as input a security parameter $\lambda$ and outputs $\{\text{params}, \text{sk}\}$.

CreGen, the credential generation algorithm, is an interactive blind signature protocol between the issuer and a user who has a set of attributes $L = \{l_1, \ldots, l_n\}$. The common input is the public parameters $\text{params}$ and a commitment $C$ to the user’s attributes $\{l_1, \ldots, l_n\}$. The user’s private input is $(l_1, \ldots, l_n)$, an opening $r$, and a message $m$; the issuer’s private input is the secret key $sk$. At the end of the protocol, the issuer outputs either “completed” or “non-completed,” while the user outputs a credential $\sigma = \{C, \tilde{\sigma}\}$, where $C$ is a fresh commitment on the same attributes and $\tilde{\sigma}$ is a signature on the pair $(m, C)$.

Verify$(\text{params}, m, \sigma)$ is the verification algorithm that takes as input the public parameters $\text{params}$, a message $m$, and a credential $\sigma$. It outputs 1 if the credential is valid; otherwise, it outputs 0.

An anonymous credential scheme has to satisfy two security properties: blindness and one-more unforgeability.

3.4.1. Blindness. The blindness averts the signer to view the message and the attributes he signs. More formally, the blindness can be defined using the following security game between the adversary $A$ and the challenger $C$:

(1) The adversary $A$ runs the Setup to create the public parameters $\text{params}$ and a secret key $sk$. The parameters $\text{params}$ are made public, while $A$ keeps $sk$ as his private key.

(2) $A$ outputs two attribute sets $L_0, L_1$ and two messages $m_0, m_1$.

(3) The challenger $C$ picks a random bit $b$ that is unknown to $A$. $C$ forms two commitments $C_{b_0}, C_{1-b}$ for $L_0, L_{1-b}$, respectively. After two parallel interactive credential generation protocols, $C$ obtains two credentials $\sigma_{b_0}, \sigma_{1-b}$, which are given to $A$.

(4) $A$ outputs a bit $b'$. $A$ wins the game if $b' = b$ holds. The anonymous credential scheme is said to fulfill the blindness if the advantage $\Pr[b' = b] - 1/2$ for any PPT adversary $A$ is negligible.
3.4.2. One-More Unforgeability. The one-more unforgeability prevents the user from producing more signatures than requested. More formally, the one-more unforgeability can be defined using the following security game between the adversary \( \mathcal{A} \) and the challenger \( \mathcal{C} \):

1. The challenger \( \mathcal{C} \) runs the Setup, and the parameters \( pp \) are given to the adversary \( \mathcal{A} \), while \( sk \) is kept secret by \( \mathcal{C} \).
2. \( \mathcal{A} \) engages in polynomially many interactive credential generation protocols with \( \mathcal{C} \). For every execution, \( \mathcal{A} \) picks an attribute set \( L_i \) and generates its corresponding commitment \( C_i \), then selects a message \( m_i \), and finally outputs a credential \( \sigma_i \).
3. \( \mathcal{A} \) outputs a list of message and credential pairs \( \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\} \).
4. \( \mathcal{A} \) chooses an attribute set and message pair \((L^*, m^*)\) not contained in the queried set \( \{L_i, m_i\} \) and outputs \((m^*, \sigma^*)\).

\( \mathcal{A} \) wins the game if \( \text{Verify}(\text{params}, m^*, \sigma^*) = 1 \). The anonymous credential scheme is said to fulfill the one-more unforgeability if the advantage \( \Pr[\mathcal{A} \text{ wins}] \) for any PPT adversary \( \mathcal{A} \) is negligible.

3.5. Zero-Knowledge Proof. Zero-knowledge proofs are fundamental cryptographic primitives that allow a prover to convince a verifier that the former knows some secret information without leaking any additional information. Let \( R \) be an efficiently computable binary relation that includes pairs of form \((x, w)\) where \( x \) is a statement and \( w \) is a witness. Let \( \mathcal{L} \) be the language associated with the relation \( R \), i.e., \( \mathcal{L} = \{x \mid \exists w \text{ s.t. } R(x, w) = 1\} \). A zero-knowledge proof for \( \mathcal{L} \) is to let a prover convince a verifier that \( x \in \mathcal{L} \) for a common input \( x \) without revealing \( w \) [33].

The zero-knowledge succinct noninteractive arguments of knowledge (zk-SNARKs) are noninteractive proof systems, where succinct means that the proof is required to be very short and easy to verify. A zk-SNARK for an NP language \( \mathcal{L} \) consists of three algorithms (Setup, Prover, Verifier) defined as follows:

- **Setup** \((1^\lambda, \mathcal{L}) \) is the setup algorithm that outputs a common reference string \( pp \), given a security parameter \( \lambda \) and a description of the language \( \mathcal{L} \).
- **Prover** \((x, w, pp) \) is the proving algorithm that takes as input a statement \( x \), a witness \( w \), and the common reference string \( pp \). It outputs a noninteractive proof \( \eta \).
- **Verifier** \((x, \eta, pp) \) is the verification algorithm that takes as input a public \( x \), a proof \( \eta \), and the common reference string \( pp \). It outputs 1 if \( \eta \) is a valid proof such that \( x \in \mathcal{L} \) or 0 if the proof is rejected.

A zk-SNARK should satisfy the following properties.

3.5.1. Completeness. For each pair \((x, w) \in R\), we have

\[
\Pr[pp \leftarrow \text{Setup}(1^\lambda, \mathcal{L}), \eta \leftarrow \text{Prover}(x, w, pp) \mid \text{Verifier}(x, \eta, pp) = 1] = 1
\]

(1)

3.5.2. Soundness. For any PPT adversary \( \mathcal{A} \),

\[
\Pr[pp \leftarrow \text{Setup}(1^\lambda, \mathcal{L}), x \notin \mathcal{L}, (x, \eta) \leftarrow \mathcal{A}(pp) \mid \text{Verifier}(x, \eta, pp) = 1] = \text{negl}(\lambda)
\]

(2)

3.5.3. Succinctness. The size of a proof and verification time is polynomial in \( \lambda \).

3.5.4. Zero Knowledge. For any PPT adversary \( \mathcal{A} \), there exists a PPT simulator \( \mathcal{S} = (\delta_1, \delta_2) \), such that

\[
\Pr[pp \leftarrow \text{Setup}(1^\lambda, \mathcal{L}), (x, w) \in R, \delta_1(1^\lambda, \mathcal{L}) \leftarrow (x, w), \mathcal{A}(\eta) = 1] = \Pr[pp, \delta_1(1^\lambda, \mathcal{L}) \leftarrow (x, w), \delta_2(x, \text{trap}, pp, \delta_1(1^\lambda, \mathcal{L})) \leftarrow (x, w), \delta_2(1^\lambda, \mathcal{L}) \leftarrow (x, w)]
\]

(3)

where trap is the simulation trapdoor.

3.5.5. Proof of Knowledge. For any PPT adversary \( \mathcal{A} \), there exists a PPT extractor \( \mathcal{E} \), such that, for any auxiliary information \( z \), the following holds:

\[
\Pr[pp \leftarrow \text{Setup}(1^\lambda, \mathcal{L}), (x, \eta) \leftarrow \mathcal{A}(pp, z), (x, w) \notin R, \text{Verifier}(x, \eta, pp) = 1 \mid \mathcal{E}(pp, z)] = \text{negl}(\lambda)
\]

(4)

3.6. Blockchain and Smart Contract. A blockchain is a distributed and public ledger maintained by a peer-to-peer network which is running based on a predefined consensus model. In such a distributed network, numerous transactions referring to use-case data are included into a block. Several blocks connect together in a designated way to form a chain called blockchain. Once a block is confirmed and appended to the chain, these transactions forming this block cannot be changed; namely, blockchain makes it possible to construct a miniature database which ensures that the uploaded data is shared yet immutable.
The transactions are packaged together so as to build a block by a miner, who will obtain a mining reward after successfully generating a block according to a consensus mechanism. PoW is a typical protocol for achieving consensus, which is a demanding job to find an appropriate nonce so that the hash of a newly generated block can meet a specified requirement. Then, others can check its correctness when a new block is created.

Smart contract [34] serves as a self-executing program which transfers complicated procedures of an application into codes; namely, it represents how to deal with all the transactions automatically. Blockchain node runs this digital contract locally and records the execution result, which can be verified through the generated new blocks. Without relying on the role of any trusted third-party arbitrator, the faithfulness and correctness can be ensured as the protocol runs, once all the policies of an application are predeployed in the contract.

4. Traceable Attribute-Based Anonymous Authentication

Before formalizing the voting protocol, we first formulate the new cryptographic concept called traceable attribute-based anonymous authentication. The new tool is able to combine with credential generation and zero-knowledge proof technique to provide a more general authentication model.

4.1. Syntax. A traceable attribute-based anonymous authentication, TABAA, is a tuple of five algorithms (Setup, CredGen, Auth, Verify, Trace):

\[\text{Setup}(1^k) \rightarrow (mpk, msk).\] Taking as input a security parameter \(1^k\), the algorithm outputs a master public key \(mpk\) and a master secret key \(msk\).

\[\text{CredGen}.\] The credential generation algorithm is an interactive protocol between the issuer and a user who has an attribute set \(L = \{id, I_1, \ldots, I_n\}\). The common input is the master public key \(mpk\) and a commitment \(C\) to the set \(L\). The issuer’s private input is the master secret key \(msk\), and the user’s private input is a serial number \(cid\), his attribute set \(L\), and an opening \(r\). At the end of the protocol, the issuer’s output is “completed” or “non-completed,” while the user outputs an anonymous credential \(\sigma = (C, \overline{C})\), where \(C\) is a fresh commitment on the same attributes and \(\overline{C}\) is a signature on the pair \((cid, C)\).

\[\text{Auth}(M = p\|\lambda\|m, L, r, C, \sigma, mpk) \rightarrow \pi.\] It takes as input a message \(M\) consisting of an “address” \(p\|\lambda\) (i.e., an event identifier and an access policy), a payload \(m\) (in \(\lambda\)), the message to be signed is split into two parts, i.e., an address and a payload; here we use the symbol \(\|\) to connect the address and payload), an attribute set \(L\), an opening \(r\), a commitment \(C\), a credential \(\sigma\), and the master public key \(mpk\). The algorithm outputs an authentication token \(\pi\) on the message \(M\).

\[\text{Verify}(M, \pi, mpk) \rightarrow 0/1.\] Taking as input a message \(M\), an authentication token \(\pi\), and the master public key \(mpk\), the algorithm returns 1 if the authentication token is valid; otherwise, it returns 0.

\[\text{Trace}(M_1, M_2, \pi_1, \pi_2) \rightarrow \bot/id.\] Taking as input two message and authentication pairs \((M_1, \pi_1), (M_2, \pi_2)\), the algorithm returns \(id\) if \(M_1, M_2\) have a common address \(p\|\lambda\) and are authenticated by the same credential; otherwise, it returns \(\bot\).

4.1.1. Correctness. A TABAA scheme must satisfy the following:

1. Verification Correctness. An authentication token generated by an honest user should be identified as a valid authentication token with overwhelming probability.

2. Tracing Correctness. Two honestly created authentication tokens on two different messages having a common address will be linked if and only if they are generated by the same user, while the output of Trace must be the identity of this actual user.

Definition 1 (correctness). A TABAA is correct if it satisfies verification correctness and tracing correctness.

4.2. Security Definitions. Security of TABAA schemes has three aspects, namely, unforgeability, anonymity, and accountability.

4.2.1. Unforgeability. Unforgeability for TABAA schemes is defined in the following game between the challenger \(C\) and the adversary \(A\):

1. \(C\) creates a master key pair \((mpk, msk)\) and sends \(mpk\) to \(A\), while \(msk\) is kept secret by \(C\). \(A\) chooses \(n\) attribute sets \(L_1, \ldots, L_n\) and generates the corresponding commitments \(C_{i1}, \ldots, C_{in}\).

2. \(A\) runs the credential generation with \(C\). \(A\) selects an attribute set \(L_i\), submits the corresponding commitment \(C_i\) to \(C\), and finally obtains a corresponding credential \(\sigma_i\).

3. \(A\) selects a message \(M_i = p_i\|\lambda_i\|m_i\) and an attribute set \(L_i\), which is not among the set \(\{L_j\}\) during the credential query phase, and asks \(C\) to authenticate \(M_i\). \(C\) first checks whether \(L_i\) satisfies \(\lambda_i\). If yes, \(C\) then runs the CredGen to obtain a credential, runs Auth to generate a corresponding \(\pi_i\), and delivers \(\pi_i\) to \(A\); otherwise, \(C\) aborts the protocol.

4. \(A\) chooses a message \(M^* = p^*\|\lambda^*\|m^*\) and an attribute set \(L^*\) and produces a corresponding \(\pi^*\).

\(A\) wins the game if

1. \(\pi^*\) is not among the set \(\{\pi_i\}\) during the credential query phase
(2) $M^*$ is not among the set $\{M_i\}$ during the authentication query phase

(3) Verify $(M^*, \pi^*, mpk) = 1$

We denote by

\[
\text{Adv}_{\mathcal{A}}^{\text{Unf}}(\lambda) = \Pr[\mathcal{A} \text{ wins the game}] \tag{5}
\]

the success probability of adversary $\mathcal{A}$ in winning the unforgeability game.

Definition 2 (unforgeability). A TABAA scheme is unforgeable if, for all PPT adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{Unf}}(\lambda)$ is negligible.

4.2.2. Full Anonymity. Full anonymity for TABAA schemes is defined in the following game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$:

(1) $\mathcal{A}$ creates a master key pair $(mpk, msk)$. The master public key $mpk$ is given to the $\mathcal{C}$, while $msk$ is kept secret by $\mathcal{A}$.

(2) $\mathcal{C}$ runs the credential generation with $\mathcal{A}$. $\mathcal{C}$ chooses two attribute sets $L_0, L_1$ and generates two commitments $C_{00}, C_{11}$. Then, $\mathcal{C}$ submits $C_{00}, C_{11}$ to $\mathcal{A}$ and finally obtains $C_{01}, C_{10}$.

(3) $\mathcal{A}$ selects a message $M_i = p_i \| \lambda_i \| m_i$, and a bit $j \in \{0, 1\}$ corresponding to $C_j$ and asks $\mathcal{C}$ to authenticate $M_i$ by using $C_j$. However, all the $q$ queried messages cannot have a common address. If $L_0, L_1$ can both satisfy the access policy $\lambda_i$, $\mathcal{C}$ will generate a corresponding $\pi_i$ and delivers it to $\mathcal{A}$.

(4) $\mathcal{A}$ chooses a message $M^* = p^* \| \lambda^* \| m^*$, which is not among the set $\{M_i\}$ and cannot have a common address (i.e., $p_i \| \lambda_i$) with $q$ queried messages during the authentication query phase, and sends $M^*$ to $\mathcal{C}$. $\mathcal{C}$ picks a random bit $b \in \{0, 1\}$, uses $L_b, C_{b1}, \sigma_b$ to generate a corresponding $\pi^*$, and delivers $\pi^*$ to $\mathcal{A}$.

(5) $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. We denote by

\[
\text{Adv}_{\mathcal{A}}^{\text{Acc}}(\lambda) = \Pr[\mathcal{A} \text{ wins the game}] \tag{6}
\]

the success probability of adversary $\mathcal{A}$ in winning the accountability game.

Definition 4 (accountability). A TABAA scheme is accountable if, for all PPT adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{Acc}}(\lambda)$ is negligible.

Summarizing, we have the following.

Definition 5 (security). A TABAA scheme is secure if it is correct, unforgeable, anonymous, and accountable.

5. Our Traceable Attribute-Based Anonymous Authentication

The basic idea of TABAA is to use an anonymous credential scheme and a zk-SNARK scheme to achieve an authentication scheme with full anonymity while holding accountability. A user with a set of attributes obtains an attribute-based credential without revealing the real values of its attributes, i.e., registration anonymity. With the credential, the user can authenticate a message and generate a corresponding authentication token anonymously, i.e., authentication anonymity. However, if the user attempts to authentication two different messages possessing an identical event, its identity will be disclosed publicly.

5.1. Generic Construction. Let $\text{Cre} = (\text{Cre.Setup}, \text{Cre.CredGen}, \text{Cre.Verify})$ denote an anonymous credential scheme, $\text{ZK} = (\text{ZK.Setup}, \text{ZK.Prover}, \text{ZK.Verifier})$ be the zk-SNARK protocol, $\mathcal{F}$ be the space of attribute, and $L = \{id, l_1, \ldots, l_n\}$ be an attribute set from $\mathcal{F}$. Let $\lambda$ denote an attribute-based access policy and $\delta$ be a subset of set $L$, i.e., $\delta \subseteq L$. We also denote by $\lambda(\delta) = 1$ the case that $\delta$ satisfies $\lambda$.
and by $A(\delta) = 0$ otherwise. The construction of our TABAA scheme is as follows:

Setup $(L^1, L^2)$. Given a security parameter $1^\lambda$ and a language $L$ depicted in the following authentication algorithm, do the following:

1. Run $\text{Cre.Setup}(1^\lambda)$ to generate the anonymous credential parameters $pk$ and the secret key $sk$
2. Run $\text{ZK.Setup}(1^\lambda, L^2)$ to obtain the public parameters $pp$ for the zk-SNARK protocol
3. Select two hash functions $\mathcal{H}_1 : \{0, 1\}^* \rightarrow \mathcal{H}, \mathcal{H}_2 : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathcal{H}$
4. Output the public parameters $mpk = (pk, pp, \mathcal{H}_1, \mathcal{H}_2)$, but the master secret key $sk = (sk)$ is kept secretly

$\text{CredGen}$. The anonymous credential generation algorithm is an interactive protocol between the issuer and a user with an attribute set $C_i$. Run $\text{Cre.CredGen}$ algorithm. A user first forms a commitment $\Sigma_i$ by correctness of the hash function.

Given a message $M = \{A\}m$ that consists of an address $A$ and a payload $m$, this algorithm works as follows:

1. Compute a linking tag $t_1 = \mathcal{H}_1(p||A, L_1)$ and a tracing tag $t_2 = \mathcal{H}_2(p||A, id) + m \cdot id$.
2. Let $x = (M, t_1, t_2, mpk)$ be a common knowledge and $u = (L_i, r_i, C_i, \sigma_i)$ be a private witness. Call $\text{Cre.Verify}$ and $\text{ZK.Verify}$ for the language $L = \{x = (M, t_1, t_2, mpk) \exists \omega = (L_i, r_i, C_i, \sigma_i) \text{ s.t. } t_1 = \mathcal{H}_1(p||A, L_1) \land t_2 = \mathcal{H}_2(p||A, id) + m \cdot id \land C_i = \text{Commit}(pk, L_i, r_i) \land \text{Cre.Verify}(pk, \omega, \sigma_i) = 1 \land \delta_i \leq L_i\}$, where $\text{Cre.Verify}$ validates if the credential $\sigma_i$ corresponding to the commitment $C_i$ is valid; the set $\delta_i \leq L_i$ should be verified if it satisfies the access policy $A$. Then, $\text{ZK.Verify}(x, u, pp)$ produces a proof $\eta$.
3. Output an authentication token $\pi = (t_1, t_2, \eta)$.

$\text{Verify}(M, \pi, mpk)$. Taking as input a message $M_i$, an authentication token $\pi$, and the public parameters $mpk$, run $\text{ZK.Verify}(x, \pi, pp)$ and output either 0 or 1 for invalid or valid. 

$\text{Trace(M}_1, M_2, \pi_1, \pi_2)$. Taking as input two messages $M_1, M_2$ and two valid authentication tokens $\pi_1 = (t_1, t_2, \eta_1), \pi_2 = (t_1', t_2', \eta_2)$, the algorithm checks if $t_1 = t_1'$: If yes, it outputs $id$; otherwise, it returns $\bot$.

5.2. Correctness and Security Analysis. We verify the correctness of our construction.

Given a key pair $(pk, sk)$ created by $\text{Cre.Setup}(1^\lambda)$, the public parameters $pp$ produced by $\text{ZK.Setup}(1^\lambda, L^2)$, two secure hash functions $\mathcal{H}_1, \mathcal{H}_2$, and a set of attributes $L_i = \{id_1, \ldots, id_n\}$, the correctness is reflected in two aspects:

1. Verification Correctness. If the attributes in $\delta_i \leq L_i$ satisfy the access policy $A$, we have $A(\delta_i) = 1$. Since a commitment $C_i$ is formed by $\text{Commit}(pk, L_i, r_i)$, we have $C_i = \text{Commit}(pk, L_i, r_i)$ by correctness of the commitment scheme $C$. Since a credential $\sigma_i$ is generated by $\text{Cre.CredGen}$, we have $\text{Cre.Verify}(pk, cid, \sigma_i) = 1$ by correctness of the anonymous credential scheme Cre. Since a proof $\eta$ is created by $\text{ZK.Verify}(x, u, pp)$, $\eta$ can pass the authentication verification from $\text{ZK.Verify}(x, \pi, pp)$ by correctness of the zk-SNARK protocol $ZK$.

2. Tracing Correctness. Let two messages $M_1, M_2$ share a common address $p||A$; we can run $\text{Auth}(M_1, L_i, r_i, C_i, \sigma_i, mpk)$ to obtain $\pi_1, \pi_2$ such that they can pass the authentication verification test from $\text{Verify}(M_i, \pi_i, mpk)$ for $i = 1, 2$, but they must be linked because the tag $t_1$ is computed through $\mathcal{H}_1(p||A, L_i)$ using the same attribute set $L_i$. Since $t_2 = \mathcal{H}(p||A, id) + m \cdot id$, there must be an identical attribute $id$ in $\pi_1, \pi_2$ such that the $id$ can be inferred by correctness of the hash function.

Next, we state some theorems on the security of our construction.

Theorem 1 (unforgeability). Assume that the anonymous credential scheme $\text{Cre}$ is unforgeable and the zk-SNARK protocol $ZK$ is proof of knowledge; then, the TABAA scheme is unforgeable in the random oracle model.

Proof. Let $A$ be an adversary who breaks the unforgeability property of our TABAA scheme with nonnegligible probability. Using $A$, we construct an algorithm $B$ against the unforgeability of $\text{Cre}$.

$B$ receives $pk$ from its $\text{Cre}$. challenger, denoted by $\mathcal{C}$, who runs the $\text{Cre.Setup}$ algorithm. $B$ runs $\text{ZK.Setup}$ and chooses two hash functions $\mathcal{H}_1, \mathcal{H}_2$; then, $B$ sends $mpk = (pk, pp, \mathcal{H}_1, \mathcal{H}_2)$ to $A$. $A$ chooses $n$ attribute sets $L_1, \ldots, L_n$ and generates the corresponding commitments $C_1, \ldots, C_n$. $A$ selects an attribute set $L_i$ and sends the corresponding commitment $C_i$ to $B$, and $B$ will transmit it to $\mathcal{C}$. $B$ will act as an intermediary for delivering messages during the interaction between $A$ and $\mathcal{C}$. Finally, $A$ will obtain an anonymous credential $\sigma_i$. Next, $A$ chooses a message $M_i = p||A_i||m_i$ and an attribute set $L_i$, which is not among the set
\{L_b\}. A sends \(M_1, L_1\) to \(\mathcal{B}\), and \(\mathcal{B}\) will invoke the simulator algorithm of zk-SNARK to create a corresponding \(\pi_1\), and deliver \(\pi_1\) to \(\mathcal{A}\). Eventually, \(\mathcal{A}\) selects an attribute set \(L^*\) and outputs his forgery \((M^*, \pi^*)\).

Owing to the proof of knowledge property of the zk-SNARK protocol, \(\mathcal{B}\) can call the extractor algorithm of zk-SNARK to extract the credential \(\pi^*\) from \(\pi^*\), and output a forgery \((C^*, \sigma^*)\). However, \(L^*\) was never queried before, which means that \(\mathcal{A}\) forges a credential. Since we assume that the anonymous credential scheme \(\mathcal{C}\) is unforgeable, \(\mathcal{A}\) can break the unforgeability of our TABAA scheme with negligible probability.

\[\square\]

**Theorem 2** (anonymity). Assume that the anonymous credential scheme \(\mathcal{C}\) is blind and the zk-SNARK protocol \(\mathcal{ZK}\) is zero knowledge; then, the TABAA scheme is anonymous in the random oracle model.

**Proof.** Suppose that there exists an adversary \(\mathcal{A}\) who can break the anonymity property of our TABAA scheme with nonnegligible probability; we construct an algorithm \(\mathcal{B}\) against the zero-knowledge property of \(\mathcal{ZK}\) with nonnegligible probability.

\(\mathcal{A}\) runs the \(\text{Cre.Setup}\) algorithm to create \((pk, sk)\) and sends \(pk\) to \(\mathcal{B}\). \(\mathcal{B}\) runs \(\text{zk.Setup}\) to create two hash functions \(H_1, H_2\), and sends \(mpk = (pk, pp, H_1, H_2)\) to \(\mathcal{A}\). Then, \(\mathcal{C}\) chooses \(n\) attribute sets \(L_1,\ldots,L_n\) and sends them to \(\mathcal{A}\). \(\mathcal{A}\) runs the credential generation \(\text{CredGen}\) algorithm with \(\mathcal{C}\). Then, \(\mathcal{C}\) will obtain the two corresponding credentials \(\sigma_1, \sigma_2\). Next, \(\mathcal{A}\) selects a message \(M_i = p_i || \pi_i || m_i\) and a bit \(j \in \{1, 2\}\) corresponding to \(C_j\) and sends \((M_i, j)\) to \(\mathcal{B}\). However, all the \(q\) queried messages cannot have a common address. \(\mathcal{B}\) transmits \((M_i, j)\) to \(\mathcal{C}\). If \(L_1, L_2\) both satisfy the access policy \(H_{i1}, H_{i2}\), \(\mathcal{C}\) will generate a corresponding \(\pi_i\) and deliver it to \(\mathcal{B}\), and \(\mathcal{B}\) will send \(\pi_i\) to \(\mathcal{A}\). Then, \(\mathcal{A}\) chooses a message \(M^* = p^* || \pi^* || m^*\), which is not among the set \(\{M_i\}\) and cannot have a common address with \(q\) queried messages during the authentication query phase, and sends \(M^*\) to \(\mathcal{B}\), and \(\mathcal{B}\) sends \(M^*\) to \(\mathcal{C}\). \(\mathcal{C}\) picks a random bit \(b \in \{1, 2\}\), uses \(L_{ib}, C_{ib}\), \(\pi_b\) to generate a corresponding \(\pi^*\), and sends \(\pi^*\) to \(\mathcal{B}\), and \(\mathcal{B}\) transmits \(\pi^*\) to \(\mathcal{A}\). Eventually, \(\mathcal{A}\) outputs a guess \(b \in \{1, 2\}\).

Owing to the blindness of the anonymous credential scheme \(\mathcal{C}\), \(\mathcal{A}\) cannot obtain any information about \(L_{ib}\) in the credential query phase; due to the zero-knowledge property of zk-SNARK protocol \(\mathcal{ZK}\) and since \(H_1\) and \(H_2\) are random oracles, \(\pi_b\) reveals nothing about \(L_{ib}\); moreover, \(\pi^*\) is similar to the one in \(\pi_i\) in the authentication query phases; thus, there is also no leakage of \(L_{ib}\). Therefore, \(\mathcal{A}\) cannot get information about \(L_{ib}\) in the above three phases. The successful probability that \(\mathcal{A}\) can guess \(b\) correctly is always 1/2.

\[\square\]

**Theorem 3.** (accountability). Assume that the zk-SNARK protocol \(\mathcal{ZK}\) is proof of knowledge; then, the TABAA scheme is accountable in the random oracle model.

**Proof.** Suppose that there exists an adversary \(\mathcal{A}\) who can break the accountability property of our TABAA scheme with nonnegligible probability; we can construct an algorithm \(\mathcal{B}\) against the property of \(\mathcal{ZK}\) with nonnegligible probability.

\(\mathcal{B}\) receives \(pk\) from its challenger, denoted by \(\mathcal{C}\), who runs the \(\text{Cre.Setup}\) algorithm to create \((pk, sk)\). \(\mathcal{B}\) runs \(\text{zk.Setup}\) to create \(pp\), chooses two hash functions \(H_1, H_2\), and sends \(mpk = (pk, pp, H_1, H_2)\) to \(\mathcal{A}\). Then, \(\mathcal{C}\) chooses \(n\) attribute sets \(L_1,\ldots,L_n\) and sends them to \(\mathcal{A}\). \(\mathcal{A}\) creates his attribute set \(L = \{id, L_1,\ldots,L_n\}\) and runs the credential generation protocol. At first, \(\mathcal{A}\) forms a commitment \(C\) with \(L\) and sends \(C\) to \(\mathcal{B}\), and \(\mathcal{B}\) transmits \(C\) to \(\mathcal{C}\); then, \(\mathcal{A}\) will obtain a credential \(\sigma\) after the interactive credential generation protocol. Next, \(\mathcal{A}\) submits a commitment \(C_j\) to \(\mathcal{B}\), \(\mathcal{B}\) will transmit it to \(\mathcal{C}\), and finally \(\mathcal{A}\) will obtain a corresponding credential \(\sigma_j\). \(\mathcal{C}\) submits \(\pi_i\) to \(\mathcal{A}\), and \(\mathcal{A}\) sends \(\pi_j\) to \(\mathcal{B}\). Finally, \(\mathcal{A}\) chooses two messages \(M_1 = p_1^* || \pi_1^* || m_1\) and \(M_2 = p_2^* || \pi_2^* || m_2\) sharing a common address \(p^* || \pi^*\), and \(\mathcal{C}\) gives the corresponding \((\pi_1, \pi_2, (M_1, \pi_1), (M_2, \pi_2))\).

If \(\pi_1, \pi_2\) can pass verification by \(\text{Verify}\), \(\mathcal{A}\)‘s identity can be extracted by calling the extractor algorithm of zk-SNARK, because the credential \(\mathcal{A}\) used to generate \(\pi_1\) and \(\pi_2\) must be the credential \(\sigma\) registered before, and \(\mathcal{A}\) only has this credential, or else \(\mathcal{A}\) forged a credential. However, we have proved that our TABAA scheme is unforgeable. Therefore, if \(\mathcal{A}\) can give a valid authentication, he must use the credential which is registered before, and this credential contains the attribute information. We can extract the attribute set \(\mathcal{L}\) according to the proof of knowledge property of zk-SNARK. Eventually, \(\mathcal{A}\)‘s identity \(id\) is revealed.

\[\square\]

5.3. Performance Analysis. We instantiate the anonymous credential scheme \(\mathcal{C}\) by the blind Schnorr signature [36] and the Pedersen commitment [37] and use the zk-SNARK scheme proposed by Ben-Sasson et al. [38] to instantiate the \(\mathcal{ZK}\) scheme. We exploit the libsnark library [39], which achieves the zk-SNARK in C++, and implement our TABAA scheme on a computer with AMD Ryzen 5 3600 6-core processor running Ubuntu 20.04. We analyze the performance of TABAA as follows.

We select some recent anonymous authentication schemes holding linking-and-tracing, where [40] only achieves the basic anonymity in the authentication phase, while [7] further ensures full anonymity. Note that we consider an anonymous signature generation as the authentication on a message. To authenticate a message, we need to generate a noninteractive zero-knowledge proof by using the libsnark; thus, we list the running time in authentication and verification, corresponding to the ZK.Prover and ZK.Verifier, respectively. Although it takes a slightly longer time to create an authentication compared to the other two schemes, the verification time is only 5.40 ms.
We exploit the zk-SNARK to generate a proof corresponding to an authentication on a message, and the size of the authentication is 0.12 KB. An anonymous authentication consists of a message and a corresponding authentication token; thus, the storage cost and the communication cost are the same, i.e., 0.21 KB. From Table 2, we can see that our scheme has a shorter verification time, a smaller authentication size, and a lower storage and communication cost than others. The simulation result shows that our TABAA scheme is feasible and efficient.

6. Anonymous and Traceable Voting on Blockchain

As one of the most challenging but attractive cryptographic protocol issues, e-voting can be widely adopted to large-scale national election and small-scale boardroom voting and classroom voting. Cryptographic techniques such as signature schemes and encryption schemes are usually exploited to implement privacy-preserving voting systems. In this section, more descriptions about voting model and its security requirements are given as follows. Based on the system model, we also elaborate the details and analysis of our voting scheme.

Electronic voting, which provides a convenient manner for citizens to exercise their rights in a modern democracy, gains considerable attention and comes into widespread usage compared to traditional voting, which always suffers from inefficiency and unintentional errors [41]. While serving as a database for aggregating all the ballots from voters, centralized e-voting platform is prone to privacy breach, data loss, data modification, and single point failure. Bulletin board-based voting system can be regarded as a public broadcast channel, such that all the voting-related information is available and anyone cannot remove the data that has been written on the board [42], but one of the major concerns is whether the data appearing on the board is trusted or not [43]. Instead, blockchain acting as a transparent, immutable, and shared ledger can be considered as a public bulletin board that consists of a range of blocks. The blockchain is jointly maintained by a peer-to-peer network, which consists of multiple nodes whose task is to package transactions to a block that will be validated by these nodes according to a designated consensus mechanism. This guarantees powerful credibility when delivering messages through a distrusted network. Moreover, the smart contract works as a self-executing program that faithfully follows orders and carries out all the required calculations. We mainly review some typical e-voting schemes as follows.

6.1. Bulletin-Based Voting. Various cryptographic techniques are widely exploited for voting through being combined with public bulletin board. Helios [44] was based on mixnet which is for the sake of shuffling the order of ballots so as to provide anonymity protection. However, multiple mixers requiring computational effort to prove the correctness make this kind of voting generally inefficient [42]. Homomorphic encryption realizes operating on multiple ciphertexts without decrypting them and is extensively used in Paillier encryption-based voting [45] and ElGamal encryption-based voting [46]. However, the main shortcoming is that these schemes need to do a mass of computation. In addition, signature schemes such as blind signature, group signature, and ring signature are also incorporated into voting systems. Blind signature-based voting [47] allows signing an encrypted message without knowing its content, but requiring anonymous channels is the major problem. Ring signature-based voting [42] aims to achieve strong anonymity by allowing a member to sign a message on behalf of a group, but it cannot trace the malicious voter. Secret-sharing-based voting [48] increases the reliability and confidentiality [49], but it involves multiple parties. The above-mentioned schemes are restricted by a centralized bulletin, which may fail to resist single point of failure and cannot balance the anonymity and accountability.

6.2. Blockchain-Based Voting. Numerous kinds of blockchain-based voting systems have been proposed to adapt for distinct scenarios. The first type enables exploiting the cryptocurrency. In [50], voters make use of the bitcoin transaction to participate in voting while preserving the privacy of ballot with the help of a random number. Some voting systems based on Zerocoin and Zerocash are also leveraged to enhance anonymity protection in [51, 52], respectively, but it is unpractical to use the cryptocurrency to vote. The second type is to achieve the function of self-tallying. In [53], the authors implement a self-tallying protocol with the aid of smart contract and attempt to ensure the maximum voter privacy unless collusion with all the other voters happens. In [54], the voting system is inspired by the score voting that enables voters to allocate points to candidates within a fixed total number of points. In [50], the protocol can also be deemed as self-tallying due to the special skill in setting the sum of all the randomness, but the study in [54] has multiple candidates and the study in [50, 53] only support two. A major issue of self-tallying is that the absence of one of all voters will lead to the failure of tallying. The third type only regards blockchain as a ballot box. The authors of [54–57] only regard blockchain as a trusted bulletin to collect all the ballots. All above schemes cannot ensure full anonymity and support public traceability.

6.3. Accountable Voting. Yu et al. [43] designed a decentralized voting system that supports double voting detection, such that any two ballots from the same voter can be linked. Lyu et al. [58] used linkable ring signature to hide each voter’s identity while avoiding double voting. Dimitriou [59] presented a coercion-free and universally verifiable voting, which introduces a novel way to prevent double voting. Although double-spending can be avoided, this scheme cannot achieve stronger accountability to trace malicious voter. In [60], it is possible to link voter’s identity to the corresponding ballot by means of the cooperation of multiple parities. In [61], the traceability is realized by allowing the administrator to trace the owner of a ballot but involves
several trusted parties. Moreover, Li et al. [62] presented a new type of voting scheme that balances the anonymity and accountability; the linkability feature is realized to detect double voting while the role of supervision authority is introduced to trace the illegal voter. The authors of [60–62] must draw support from the trusted parity to accomplish the traceability, but our work ensures the public traceability without cooperation with any trusted party.

It becomes more appealing to construct a decentralized voting system on blockchain. Unfortunately, we are also facing more difficult challenges that need to be urgently addressed. One noticeable issue is the transparency of blockchain, on which all the transmitted data is accessible and visible to the public, yet this gives rise to the pressing matter of contravening data privacy. What is more, the transaction history related to a user is recorded on the blockchain; thus, it may disclose the personal information about this user and thus violate user’s privacy.

Additionally, many applications, such as e-auction and e-voting, usually need user engagement before registration, and require users to give only one submission. In a real-life scenario, a user may just finish a voting and then he will join in another new voting or other auctions, but a registered certificate is unsupported in other different activities. In this process, four urgent issues arise. (i) Each user should register multiple times for different activities; apparently, it is too cumbersome and unnecessary for users. (ii) A considerable amount of sensitive information may be transparently revealed without any protection during the registration process, and an attacker may intercept and capture this information such that the user’s privacy is compromised. (iii) Efficient access mechanism should be established so as to exclude unqualified users. (iv) How double submission happening in one round of the application can be easily detected? And how the privacy and accountability can be balanced?

We summarize the comparison between our scheme and other blockchain-based voting schemes in Table 3. So far, none of the existing research has addressed all the above fundamental challenges simultaneously. In this paper, we design a traceable attribute-based anonymous voting scheme to overcome these difficult issues.

We propose the first blockchain-based anonymous and traceable voting system with enhanced privacy preservation. Our voting system shows powerful superiority in ensuring user privacy and in providing a type of reusable credential which enables reducing repeated registration. We also set an attribute-based access policy to specify who can be involved in voting, and we further achieve the public traceability to track the double-voting user so as to ensure the correctness of election result. By comparison with the above blockchain-based voting schemes in Table 3, our research is superior to the prior works by providing stronger privacy protection and further supporting other powerful functionalities, including access policy and public accountability.

### 6.4. Voting Model
As shown in Figure 1, four roles are involved in the model of voting: a certificate authority (CA), an election committee (EC), many voters \((V_1, \ldots, V_n)\), and a voting platform:

1. **Certificate authority** acts as an identity verifier to validate whether the registration information corresponds to a user’s attributes so as to prevent malicious users, and meantime issues an anonymous credential to the user. This authority plays a critical role in verifying and managing identities of users by binding their own attributes to a particular certificate, respectively, when registering.

2. **Election committee** plays the role of an event organizer who can post a voting activity to collect plenty of ballots cast by voters. In order to limit a specific group of users who should not have the suffrage, a particular limited condition can be designated by this role to determine who can be eligible to engage in voting; e.g., the age should be eighteen or over. To ensure that the election progress is timely and available, a deadline should be also set before the election starts.

3. **A voter** has a set of attributes that are different from others and should register a certificate authority to obtain an anonymous credential without sacrificing his personal information. Then, he can access this voting activity and cast a ballot unless he satisfies the limitation condition.

4. **Voting platform** is a trusted party which serves as a database to store detailed data related to the voting activity and to collect ballots from voters. In particular, it is a decentralized platform which is jointly maintained by a group of network peers.

In our scheme, we assume that each user should register to receive an anonymous credential before employing services on the voting platform, then the election committee posts a voting activity, and voters can submit ballots. We also require that each voter is only allowed to cast once in a voting activity.

Suppose that a registered election committee releases a voting activity with a concrete restriction (i.e., an access control). Any eligible voter who has been certified and meets the limitation will have the right to cast a ballot. Then, verification process will be launched to check the validity of
each voter’s identity and the corresponding ballot. Finally, all the valid ballots which have been confirmed and passed the above verification will be counted to determine who wins in the election.

Particularly, the restriction could be an age-based or region-based requirement, such as a state election, or it could be confined to a particular identity-restricted demand, for instance, a boardroom or classroom voting. Different kinds of constraints are essential to meet various forms of elections according to the concrete requirements, which are extendable and can be straightforwardly instantiated in a simple manner. Therefore, our scheme design will pay attention to this setting in order to achieve the purpose of access control.

6.5. Security Model. The potential malicious users attempt to disturb or destroy the normal functionality and order and maximize their own profits in a spitful way. We list the basic security requirements for our voting protocol as follows:

(i) **Confidentiality.** The ballots stored in voting platform are highly sensitive data, which should be kept undisclosed and cannot leak any private information to the public, except that the election committee who needs to calculate the voting result can decrypt them. Meanwhile, other additional in-process products used for proof or authentication should also hold the confidentiality to anyone.

(ii) **Anonymity.** In general, user’s personal information is revealed in plaintext when registering a credential or can be explored and tracked by linking the submissions that this user has transmitted. Intuitively, we may need three features for user’s anonymity: (i) anonymity when executing registration of a credential; (ii) anonymity when authenticating a message; and (iii) unlinkability between a ballot and a voter. Notice that the anonymity should still be retained even if the certificate authority, the election committee, and the great majority of voters are corrupted together.

(iii) **Unforgeability.** An attacker may forge one’s identity to engage in voting instead of the voter himself, so that the former can maximize the profits. The unforgeability requires that even if a voter’s credential is exposed to the public, no one can generate a valid ballot in the name of this voter. In addition, any forged ballot will fail to pass verification.

(iv) **Verifiability.** Necessary proofs are provided to attest that the corresponding operations are performed correctly. We may require four aspects of verifiability: (i) the eligibility matching the access control can be verified, (ii) the credential can be verified, (iii) the authentication generated for a ballot can be verified, and (iv) the final election result can be verified.

(v) **Accountability.** A misbehaved voter may try to cast two or more ballots to disturb the regular rule so as to accomplish his maximum benefits. A typical case is double voting, which happens by submitting two ballots from the same voter. The public accountability aims to catch the double-voting attacker who casts ballots twice, and his identity will be exposed to the public.

6.6. Security Assumption. We assume the fundamental security to guarantee the normal running of our voting system.

6.6.1. Secure Encryption Algorithm. The security of the ballot is ensured by exploiting a secure public key encryption algorithm. Each voter encrypts his own ballot by leveraging a specified public key corresponding to a secret key, which is used to decrypt successfully the encrypted ballot. Specifically, each ballot saved as a ciphertext that will be delivered to the blockchain network in the form of a transaction.

6.6.2. Majority Honest Security. The security of voting activity on blockchain is mainly dependent on the security of the blockchain platform. Considering that the election
committee and voters can also act as miners, we suppose that the essential security of blockchain cannot be broken by the attackers; i.e., the attackers do not yet possess the vast majority of resource or capability to take charge of the blockchain network, and most of the miners are actually honest. In addition, the network is low latency; message synchronization is running properly between honest miners.

6.7. Voting Protocol. Combining the advantages of blockchain, we present the anonymous and traceable voting protocol achieved on the base of the blockchain platform. Our basic strategy is to enable a user to register an anonymous credential without privacy breach, and the user can anonymously take part in an election using this credential if he satisfies an access control. All the ballots are stored in a distributed ledger that the underlying blockchain provides. Double voting can be detected, and the identity of double-voting participants will be exposed publicly. We will simply illustrate the main ideas of our protocol.

6.7.1. Anonymous Registration. To obtain an anonymous credential without sacrificing personal privacy, each voter should first commit his attributes to produce a commitment, and provide a corresponding zero-knowledge proof which indicates that he has indeed committed to the correct attributes. After receiving a commitment and its proof, the certificate authority carries out a credential issuing protocol with the voter. As a result, the voter will obtain an anonymous credential with attributes. When he needs to attest that he owns a credential with the desired attributes (e.g., age, region), he can provide the newly generated credential and prove in zero knowledge that this credential indeed contains the required attributes.

6.7.2. Anonymous Authentication. In order to ensure that the ballots are secretly generated by voters themselves, every voter must authenticate a ballot using his private information, but the whole process reveals nothing about the ballot or his personal information to the public. More specifically, zk-SNARK is leveraged to attest that the voter does indeed follow the protocol while revealing nothing except this fact. The result of authentication process can be verified by the public, but no efficient manners can deduce his identity if the voter does this authentication procedure only once.

6.7.3. Tracing Double Voters. Locating an honest voter who submits a ballot only once is unnecessary; in fact, this operation violates user’s privacy. However, it becomes vital to trace a certain voter who casts ballots twice. Linking any two authentication tags corresponding to two ballots so as to detect double voting in a subtle manner, we can recover the double voter’s identity when launching the execution of accountability. To support the public traceability, we enrich the functionality such that if a voter authenticates twice, his identity will be further revealed to the public.

As mentioned before, to avoid deanonymization in the blockchain network, we let EC and each voter create a one-time blockchain address for themselves. We also specify an access policy to restrict unqualified participants’ access to the voting. Next, we formalize a concrete scheme of the privacy-preserving voting with public traceability on top of blockchain.

Let $Ψ = (Ψ.Setup, Ψ.CredGen, Ψ.Auth, Ψ.Verify, Ψ.Trace)$ be a traceable attribute-based anonymous authentication scheme and $Π = (Π.KeyGen, Π.Enc, Π.Dec)$ be a public key encryption scheme that is semantically secure. The voting scheme consists of seven phases; details are depicted as follows:

**Step 1: Setup.** All necessary parameters are generated for initialization:

1. CA starts to run $Ψ.Setup$ to generate his public and secret key pair $(mpk, msk)$
2. CA puts $mpk$ to a transaction and sends the transaction to the blockchain network

**Step 2: Registration.** Everyone registers at CA to obtain an anonymous credential tied to his attributes:

1. CA designates the attribute types and the corresponding order to define a specific form of sequence of attributes $\{id_1, l_1, \ldots, l_n\}$. A possible setting is to require the attribute $id$ as user’s real identity.
2. $EC$ (or $V_i$) forms a commitment $C_{EC}$ (or $C_i$) with $r_{EC}$ (or $r_i$) for the attribute set $L_{EC}$ (or $L_i$), runs $Ψ.CredGen$ with CA, and obtains an anonymous credential $σ_{EC}$ (or $σ_i$).

**Step 3: Publication.** EC releases a voting activity with an access policy $A$:

1. When $EC$ prepares to post a voting activity, he creates a new blockchain account address $α_{EC}$; selects a unique voting id denoted as $vid$, the list of candidates represented as $\{ID_1, \ldots, ID_n\}$, and the deadline $dl$; and runs $Π.KeyGen$ to create a key pair $(epk, esk)$, where $epk$ is only used to encrypt the ballot.
2. More importantly, $EC$ sets an access policy $A$ requiring that only qualified participants who satisfy the access restriction are able to engage in voting.
3. $EC$ authenticates $vid∥A∥α_{EC}$ to convince voters that the voting activity numbered $vid$ definitely originates from an election committee. Thus, he should prove the validity of this voting by generating the proof $π_{EC}$.
4. $EC$ compiles a smart contract $SC$ that contains all the related public parameters (e.g., $epk$, $dl$, candidate list), rules (e.g., the tallying mechanism $T$), and constraints (e.g., the access policy $A$) for the voting and employs the address $α_{EC}$ to transmit the proof $π_{EC}$ and the contract code to blockchain network through a transaction. Only when a newly
generated block includes contract SC, can all the voters join in this voting activity.

Step 4: Casting. Each voter starts to participate in voting:

1. When voter \( V_i \) with an attribute set \( L_i \) captures the voting numbered \( \text{vid}_i \), he first examines the content of the contact SC and judges if he is permitted to vote. Once he matches the access policy \( A \), he will create a one-time blockchain address \( a_i \) and run \( \Pi \).Enc to encrypt one candidate under the key \( epk \) to generate an encrypted ballot \( B_i \).

2. Next, he should invoke \( \Psi \).Auth to establish an authentication token \( \pi_i = \Psi \).Auth(\( \text{vid}_i \| A \| B_i \), \( L_i \), \( r_i \), \( C_i \), \( \sigma_i \), \( mpk \)), which indicates that he indeed satisfies the specified access policy and generates a valid ballot.

3. Further, he uses the address \( a_i \) to transmit a ballot \( B_i \) together with \( \pi_i \) through a transaction to the blockchain network.

Step 5: Verification. Verify the validity of each ballot from the voters:

1. SC validates the ballot’s validity by checking the validity of the authentication token \( \pi_i \). Specifically, SC only runs \( \Psi \).Verify(\( \text{vid}_i \| A \| B_i \| \pi_i \| mpk \)) to test if \( \pi_i \) is faithfully constructed.

2. SC proceeds to single out the ballots that have passed verification by \( \Psi \).Verify and will save them as a pair (\( \text{vid}_i \| A \| B_i \| \pi_i \)). However, any ballot which failed to pass the verification will be dropped by SC.

3. Meanwhile, contract SC will keep on collecting ballots till the deadline.

Step 6: Tracing. Detect double voting and trace the identity of the originator who has authenticated twice:

1. In order to further filter the double-voted ballots originating from the same voter, SC will run \( \Psi \).Trace to perform checking for the collected ballots. For any two ballots which have an identical value of tag \( t_i \) in \( \pi_i \), due to the randomness of encryption algorithm, we will have two distinct ciphertexts \( B_i \) and \( B_j \). According to the corresponding \( \pi_i \) and \( \pi_j \), SC will further deduce the identity \( id = (t_2 - t_1)/(B_i - B_j) \).

2. SC picks out all the unlinked ballots so as to guarantee that \( B_i \) is the first and validated submission from a qualified voter, whereas those linked ballots corresponding to the same voter will be dropped by SC.

Step 7: Tallying. Compute and announce the election result:

1. \( EC \) will run \( \Pi \).Dec to decrypt all the valid ballots, and calculate the election result \( R = (R_1, \ldots, R_k) \) corresponding to \( k \) candidates. Additionally, he should provide a zero-knowledge proof \( \pi_R \) to prove that he really performed the tallying correctly. The proof \( \pi_R \) for the language \( L_R = \{ \text{Param, } \overline{B}, R \exists esk, s.t., b_i = \Pi \}.Dec(esk, B_i) \land epk = F(esk) \land \bigwedge_{j=1}^{k} R_j = T(ID_j; b_1, \ldots, b_n) \} \), where Param represents all related public parameters, \( \overline{B} \) denotes the valid ballots existing in the procedure of tallying, the function \( F \) is to confirm whether two keys correspond to a consistent public and secret key pair, \( b_i \) is the plaintext of \( B_i \), \( R_j \) indicates the total number of his supporters, and \( T \) defines a tallying mechanism for each candidate.

After finishing this voting, at the same time, each voter can engage in other voting activities with different identifiers. For example, when a voter \( V_i \) captures another voting numbered \( \text{vid}_i' \), he can use his credential \( \sigma_i \) without registration again and cast a ballot after checking the related information of this voting. In this process, this credential can be used repeatedly, and his personal information cannot be revealed if he follows the voting rule. More interestingly, each user can use the only credential to join in auction, crowdsourcing, etc.


6.8.1. Correctness Analysis. Correctness is guaranteed by the following cooperation from several factors.

Obviously, blockchain serves as a public, shared, and immutable distributed ledger to avoid data tampering, and smart contract acts as a self-executing program to guarantee consistent operation of the transactions; hence, the two significant superiorities provide a secure and efficient voting environment so that voters could cast their ballots and the election committee could collect and decrypt the related ballots. Moreover, (\( \text{vid}_i \| A \| B_i \| \pi_i \)) and \( (R, \pi_R) \) stored in the blockchain also provide the verifiability to ensure that the calculation is executed properly.

Meanwhile, it achieves the correctness if and only if all participants follow the protocol based on the conditions that credential \( \sigma_i \) and \( \pi_i \) can be verified by \( \Psi \).Verify in the anonymous authentication scheme and \( epk = F(esk) \) is supported in the public key encryption scheme. Furthermore, the TABAA scheme provides the correctness of tracing by \( \Psi \).Trace.

6.8.2. Security Analysis. Our voting scheme fulfills several security properties as follows:

1. Confidentiality. All the ballots stored in the blockchain are encrypted under a certain key \( epk \) of the public key encryption scheme \( \Pi \). We assume that \( \Pi \) is semantically secure; thus, the ciphertext will not reveal any private information to the public, and any adversary without \( esk \) cannot decrypt the ballots. In addition, \( \pi_i = (t_1, t_2, \eta) \) only shows two hash values
and a corresponding proof. This ensures that all the other parties cannot obtain any useful information about the content of the ballots but could verify the correctness of the authentication by the corresponding proof.

(ii) **Anonymity.** Obviously, we achieve the basic requirement of anonymity to resist privacy disclosure in the process of registration and in the authentication transcripts. Since the feature of anonymity is ensured by the anonymous authentication scheme $\Psi$, the issuer knows no extra information (i.e., the values of the attributes) other than a commitment $C_i$ during registration, no party can tell the voter’s identity from an anonymous credential $\sigma_i$, and no authentication transcript $\pi_i$ reveals private information. Thus, no adversary can link the participant with the ballot and its authentication token due to the anonymity of our TABAA scheme. Notice that we require that all the participants use the one-time blockchain address, which prevents the adversary from linking and recognizing the participant with this address.

(iii) **Unforgeability.** Even if the commitment $C_i$ and credential $\sigma_i$ are shown to the public, any message authenticated by a forged attribute set $L'$ is no doubt to be unacceptable, since the $\Psi$ scheme is unforgeable; thus, the corresponding authentication fails to pass the verification procedure by $\Psi$. Verify. In other words, without the correct attribute set $L$, the adversary cannot fake $H_1(vid||A,L)$ and $H_2(vid||A,id)$ that are the same as the ones generated by a voter. Hence, it is impracticable for an adversary to replace a voter by forging a valid authentication token.

(iv) **Verifiability.** Anyone who is qualified to access to the voting can be verified according to the satisfied subset $\delta_i$ of $L$; thus, no unqualified voter can join in voting without being discovered. We call it voter qualification verifiability. The anonymous credential $\sigma_i$ generated by $\Psi$.CredGen can also be verified by $\Psi$.Verify; this is credential verifiability. Remarking that our TABAA scheme produces an attestation $\pi_i$ when authenticating a message, checking each authentication transcript by $\Psi$.Verify, we also implement the authentication verifiability in an elaborate way. More importantly, the final election result $R$ must be trusted and can also be verified, since tallying is accomplished by $EC$ who has to offer a proof $\pi_R$ to convince the public that each valid ballot is counted correctly; we regard it as result verifiability.

(v) **Accountability.** We would rather trace double-voting user than just link two ballots from the same voter, as the former is available to reveal the voter’s true identity and plays a positive role in limiting the behavior of double voting, while the latter cannot reach this purpose. We desire to achieve the goal by detecting and recognizing the double-voting participant owing much to the linking-and-tracing property provided by the authentication tokens, i.e., the two tags $t_1$ and $t_2$ in $\pi_i$. Therefore, we can take advantage of this accountability to resist the double-voting behavior. In this way, as long as any two authenticated ballots have a common address $vid||A$ and relate to an identical voter, whose actual identity will be revealed to the public, honest voters also hold anonymity, but misbehaving voters will be faced with identity leakage.

7. **Conclusion and Future Work**

In this paper, we propose a new notion called traceable attribute-based anonymous authentication, which achieves full anonymity when issuing credentials and authenticating messages, while holding access control to exclude unqualified users from being involved in authentication and also keeping strong accountability to locate a participant when two authenticated messages having a common address originate from the same user. More attractively, we design the first attribute-based, fully anonymous, and publicly traceable voting protocol on blockchain that aims to provide more powerful privacy protection while avoiding anonymity abuse. Our voting scheme supports a special access control by setting an attribute-based policy to exclude ineligible users, and it also can trace the double-voting users without cooperation with other trusted parties. It is easy to extend this new primitive to be decentralized by introducing multiple attribute authorities, and each authority is responsible for issuing a certificate for a certain attribute. However, the authentication scheme will be inefficient with the increasing number of attributes. Therefore, how to further design a constant-size authentication mechanism involving multiple attribute authorities is an interesting problem.

Data Availability

No data were used to support this study.

Disclosure

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Natural Science Foundation.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. U2001205, 61922036, and...
References


Security and Communication Networks


