Privacy-Protected Deletable Blockchain

XIANJI CAI, YANLI REN, AND XINPENG ZHANG

School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China

Corresponding author: Yanli Ren (renyanli@shu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant U1736120 and Grant 61572309.

ABSTRACT

A deletable blockchain has been proposed recently to change the immutability of the traditional blockchain. However, the users’ identities and transaction contents are all public in the scheme, and the public data may reveal the users’ privacy. In order to protect the privacy of the users, we propose a privacy-protected deletable blockchain based on the proof-of-space consensus mechanism, which does not rely on complex cryptographic tools or any trusted party. In order to satisfy full transparency and accountability in an anonymous environment, we use a traceable ring signature or a Pedersen commitment scheme to disclose the users’ real identities or the real transaction contents respectively according to different deletion reasons. During the deletion process, we propose a linkable multi-signature scheme, which allows multiple users to generate a valid signature by using their one-time addresses as pseudonyms to protect their identity privacy. Moreover, the proposed multi-signature scheme can link two sub-signatures if they are generated by the same malicious user. Finally, we simulate the generation and deletion process of a block under the proof-of-space consensus mechanism and give the time of generating and deleting a block. The experimental results prove the efficiency and feasibility of our proposed scheme.

INDEX TERMS

Blockchain, privacy protection, deletable, linkable multi-signature.

I. INTRODUCTION

In decentralized information technology, blockchain is a breakthrough innovation which was initially invented as the foundation for Bitcoin [1] (the first decentralized cryptocurrency which is released in 2009). A distributed database containing a continuously growing list of ordered records is the basic concept of blockchain. The decentralized nature of blockchain is supported by the consensus mechanism. There are different types of consensus mechanism including the proof-of-work (PoW) [2], [3] which is based on the computing power invested by miners, while the proof-of-stake (PoS) [4] is based on the stake held by miners and the proof-of-space (PoSpace) [5] is based on the storage capacity invested by miners.

In the traditional blockchain, the transparent transaction data will leak the users’ privacy. The malicious attacker can track the flow of transactions to reveal the real identities of the users through data mining. Ignoring those privacy issues will result in impaired users’ benefits. Many solutions have been proposed recently aimed at preserving the privacy of blockchain. Non-interactive zero-knowledge proof [6], ring signature [7], and mixing services [8] are three kinds of popular methods for protecting privacy in blockchain.

One of the core features of blockchain is immutability, which means that once the block is generated, the data put on the block cannot be changed anymore. The immutability leads to a problem that once the block is generated, the block data cannot be altered or deleted even for dangerous or useless data. However, in the present deletable blockchain scheme, the users’ identities and transaction contents are all public, and the users’ privacy will be leaked during the deletion process. To mitigate this problem, it is necessary to propose a privacy-protected deletable blockchain system.

A. RELATED WORK

The deleted or redacted block will bring the changed hash link. Most of the present redactable schemes utilize a hash mechanism called chameleon hash [9] to keep the same hash link relationship between the neighboring blocks. The first approach which uses the chameleon hash is proposed by Ateniese et al. In their proposed scheme [10], when the altered or edited block brings the changed hash link, they use the chameleon hash to calculate the collision. However, the chameleon hash demands that only the reckoner who has the trapdoor can calculate the collision and this
requires a trusted party in the blockchain system. Thus, many researchers proposed an improved redactable blockchain by sharing the trapdoor key to different users.

Kondapally et al. [11] decentralize the trapdoor through a non-linear method of secret sharing, then they reconstruct the trapdoor key through the multi-party computation (MPC) technology, which ensures the modification of a block does not depend on one user but depend on multiple ones, the edit authority is given to more parties. What’s more, they propose a new idea to use a second trapdoor key to edit a block without the agreement of the block’s generator. Nevertheless, this scheme needs to create a short term key for each block, and a key authority should be constructed to manage these keys.

However, all of the proposed schemes solve this problem on the block level. Recently, Derler et al. [12] proposed a fine-grained and transaction-level redactable blockchain. In their scheme, a new policy-based chameleon hash was proposed, the new hash allows anyone who possesses sufficient privileges to satisfy the policy can then find arbitrary collisions for a given hash value. However, this solution is limited in the permissioned setting. In the permissionless setting, when the trapdoor key is shared with different users and the collision is calculated through a multi-party computation policy, this solution will suffer from scalability issues.

Instead of using the chameleon hash, Ren [13] recently proposed a deletable blockchain which is based on the proof-of-space consensus mechanism and a threshold ring signature scheme. However, all of the block data is public in this deletable proposal and it may leak the users’ real identities and the transaction contents during the deletion process. The research on proposing a privacy-protected deletable blockchain is very important and challenging.

B. OUR CONTRIBUTIONS

Inspired by the recent work, this paper proposes a privacy-protected deletable blockchain scheme based on the proof-of-space consensus mechanism [5]. We use an anonymous protocol to conceal the transaction contents and the addresses of both parties in one transaction. Then we categorize the reasons for the transaction which applies for a deletion request. In order to satisfy full transparency and accountability in an anonymous environment, we use a traceable ring signature or a Pedersen commitment scheme to disclose the users’ real identities or the real transaction contents respectively according to different privacy protection requirements.

During the process of deleting a block, we propose a linkable digital multi-signature scheme which allows the users to generate a signature by using their one-time addresses. The proposed multi-signature scheme can link two sub-signatures if they are generated by the same malicious user. The users who come to an agreement on a deletion request generate a multi-signature jointly and the rest of the users in the blockchain accept the deletion operation if the multi-signature is valid. Specifically, we provide formal security analysis, and our proposed scheme has the following features:

- Only the transaction sender has the right to propose a deletion request due to the anonymous environment. The sender can choose to reveal the real transaction contents or his real identity as the deletion evidence.
- The identities of the transaction sender and the receiver are still concealed if the transaction sender chooses to disclose the real transaction contents.
- The real transaction contents and the identity of the transaction receiver are still private if the transaction sender chooses to reveal his real identity.

Finally, we execute several experiments to simulate the mining process and give the time of generating and deleting a block. The experimental results show that the average deletion time of a block is about twice the generation time, which proves the efficiency and feasibility of our proposed scheme.

II. PRELIMINARIES

In this section, we introduce some security protocols and basic definitions, including Ring confidential transaction (Ring CT) protocol [14], traceable ring signature [15] and “proof-of-space” [5] consensus mechanism. Finally, we introduce a complexity assumption that will be used to prove security of the proposed signature scheme in Section 3.

A. RING CT PROTOCOL

The Ring CT protocol is an anonymous protocol used in Monero which contains three algorithms, multilayered linkable spontaneous anonymous group signature (MLSAG), stealth address and Petersen commitment. These algorithms are used in our blockchain to conceal the origin and destination addresses of a transaction and the transaction contents. The details can be seen in the Cryptonote protocol [16] and the Ring CT protocol [14].

MLSAG. Let \(p_{N+M}\) be a public key which presents a set of ring members. On input \(p_{N+M}\), a secret key \(sk_m\) and a message \(m \in \{0, 1\}^*\), it outputs a signature \(\sigma\).

Stealth address. On input the receiver’s real public key \((pk_A, pk_B)\) and a random number \(r\), it outputs a one-time public address \(P\).

Petersen commitment. On input the real transaction contents \(d’\) and random data \(a\), it outputs a commitment \(C\).

B. TRACEABLE RING SIGNATURES

A traceable ring signature can reveal the identity of the real signer in a particular situation without any trusted party. A tag \(L = (issue, pk_N)\) is included in the traceable ring signature, where \(issue\) refers to an identity of some social issue and \(pk_N\) is the public key set of the ring members. We donate the public key set as \(pk_N = \{pk_1, \ldots, pk_n\}\). A traceable ring signature scheme contains the following four algorithms:

- **Gen.** On input a security parameter \(k \in N\), it outputs a public/secret-key pair \((pk, sk)\).
- **Sign.** On input a message \(m \in \{0, 1\}^*\), a tag \(L = (issue, pk_N)\), and a secret key \(sk_i\), where \(i \in N\), it outputs a signature \(\sigma\).
As shown in Figure 1, let fact to construct a structure map in a certain time. We first the probability of successful mining is higher, vice versa. power to mine blocks. The size of the disk space is larger, which is based on the storage capacity. It uses the Spacemint [5] (proof-of-space) is a new type of consensus mechanism which is based on the storage capacity. It uses the public key $pk$. To further describe the relationship between each node and highlight the structure of the graph, the tag value is set for each node as follows:

$$l_i = \text{Hash}(\mu, i, l_{p_1}, l_{p_2}, \ldots, l_{p_n}), \quad i = 1, 2, \ldots, N \quad (1)$$

In equation (1), $i$ represents the index number of a node, $\mu$ is a configurable random number, $(l_{p_1}, l_{p_2}, \ldots, l_{p_n})$ represents the forward nodes which direct to the current nodes as well as the label values of their parent nodes. In this way, each node and its parent node are linked by the tag value, and thus a directed acyclic graph based on the tag values is formed.

As shown in Figure 1, it takes a certain space to store the above structure diagram. It is easier for a user to store and recover the structure diagram if the storage space invested by the user is larger. In the case of insufficient space, space is reused to store related data, data is stored and deleted repeatedly, and time is exchanged for space. Therefore, when the time is limited, the miners with different spatial sizes have different storage capacities to store the graph with the same structure, and this results in the competition based on the storage capacity. The proof-of-space consensus mechanism was built under such a model. The structure of the blockchain which is based on the proof-of-space of consensus mechanism is shown in Figure 2.

**C. PROOF-OF-SPACE CONSENSUS MECHANISM**

Spacemint [5] (proof-of-space) is a new type of consensus mechanism which is based on the storage capacity. It uses the miner’s public key $pk$ if the two signatures are generated by the same person, where $pk \in pk_N$.

**D. COMPLEXITY ASSUMPTION**

This section introduces the elliptic curve discrete logarithm problem (ECDLP) and its complexity assumption.

**Definition 1 (ECDLP):** Let $(P, Q) \in G_q$ be a random instance, where $Q = aP$ and $a \in \mathbb{Z}_q^*$. It is hard to compute $a$ from $P$ and $Q$ through any polynomial-time bounded algorithm. Assume a polynomial time-bounded algorithm $B$ can solve the ECDLP, then the probability can be defined as $Adv^{ECDLP}_{B, G_q} = Pr[B(P, Q) = a : P, Q \in G_q; a \in \mathbb{Z}_q^*, Q = aP]$. $Adv^{ECDLP}_{B, G_q} \leq \epsilon$, for any negligible function $\epsilon$.

**III. THE PROPOSED LINKABLE DIGITAL MULTI-SIGNATURE**

In this section, a new linkable digital multi-signature (LDMS) is proposed, which allows the users to delete data jointly by generating a multi-signature with their one-time addresses. The proposed LDMS scheme will be used in the following deletable blockchain.

**A. THE PROPOSED LDMS SCHEME**

Let $\{ID_1, ID_2, \ldots, ID_n\}$ be the set of the signers and $ID_1$ be the signature collector who computes the final multi-signature by collecting all the individual signatures from the entire group signers. The following polynomial time-bounded algorithms describe the proposed LDMS scheme.
• Setup:
  Assume that \( H_i (i = 0, 1, 2) : \{0, 1\}^k \to \{0, 1\}^k \) represent three distinct and secure one-way hash functions. Let \( F_q \) be a prime filed over a \( k \)-bit prime \( q \), and \( E / F_q \) be the elliptic curve points on \( F_q \). Let \( G_2 \) be an additive elliptic curve group which is a subset of \( E / F_q \), and \( P \) be a generator of \( G_2 \). The game publishes the system’s parameter set as \( \omega = \{ H_i (i = 0, 1, 2), F_q, E / F_q, G_2, P \} \).

• Set-Real-Public/Private-Keys:
  Each signer \( ID_j \) chooses a number \( k_j \in \mathbb{R} Z_q^* \) as his real private key and computes \( K_i = k_i P \) as the corresponding real public key.

• Set-One-Time-Public/Private-Keys:
  Each signer \( ID_i \) chooses \( x_i \in \mathbb{R} Z_q^* \) and \( y_i \in \mathbb{R} Z_q^* \) randomly from all of his one-time private keys, and the corresponding public keys can be computed as \( X_i = x_i P \), \( Y_i = y_i P \). Then he computes \( q_i = y_i + k_i x \), where \( l_i = H_0 (K_i, ID_i) \). Finally, \( ID_i \) sets his one-time private key as \( s_k_i = (x_i, q_i) \), and sets his one-time public key as \( p_k_i = (X_i, Y_i) \).

• LDMS-Sign:
  During the signing process, each signer \( ID_j (1 \leq i \leq n) \) sets the one-time public key \( p_k_j = (X_i, Y_i) \) as his pseudo-identity but keeps his real public key \( K_i \) secret. Each signer \( ID_j \) does the following operations to generate a multi-signature on the message \( m \):
  1) Selects a random number \( a_i \in \mathbb{R} Z_q^* \) and computes \( A_i = a_i P \).
  2) Broadcasts \( A_i \) and \( L_i = l_i K_i \) to the entire group members \( ID_j (1 \leq z \leq n, z \neq i) \).
  3) Computes \( A_0 = \sum_{i=1}^{n} A_i, X_0 = \sum_{i=1}^{n} X_i, Y_0 = \sum_{i=1}^{n} Y_i, L_0 = \sum_{i=1}^{n} L_i \) and \( Q_0 = \sum_{i=1}^{n} Q_i \), where \( Q_i = Y_i + L_i \).
  4) Computes \( h = H_1 (A_0, X_0, m) \) and \( z = H_2 (A_0, Y_0, m) \).
  5) Computes \( S_i = z q_i + h x_i + a_i \) and sends it to the signature collector \( ID_1 \).
  6) On receiving \( S_i \) and \( S_j (1 \leq i, j \leq n, i \neq j) \), \( ID_1 \) checks if the following equation holds.
  \[
  S_i P - S_j P = z (Y_i - Y_j) + h (X_i - X_j) + A_j - A_j \tag{2}
  \]
  If the equation (2) holds, \( ID_1 \) checks if the equation (2) holds. If not, \( ID_1 \) computes \( S_0 = \sum_{i=1}^{n} S_i \) and outputs the final multi-signature \( (A_0, S_0, L_0) \) on \( m \).

• LDMS-Verify:
  The verifier does the following operations to verify the multi-signature \( (A_0, S_0, L_0) \) on \( m \):
  1) Computes \( X_0 = \sum_{i=1}^{n} X_i \), and \( Y_0 = \sum_{i=1}^{n} Y_i \).
  2) Computes \( h = H_1 (A_0, X_0, m) \) and \( z = H_2 (A_0, Y_0, m) \).
  3) Checks whether the equation \( S_0 P = z (Y_0 + L_0) + h X_0 + A_0 \) holds. The verifier accepts the multi-signature \( (A_0, S_0, L_0) \) if it holds, otherwise rejects it.

B. SECURITY ANALYSIS

Under the adaptive chosen message attack, we analyze the unforgeability of our LDMS scheme in the random oracle model [17] in this section. Under the ECDLP assumption, the proposed LDMS scheme is proved to be correct, linkable and secure against the adaptive chosen message attack through the following theorems.

Theorem 1: The proposed LDMS scheme is proved to be correct and consistent.

Proof: Since we have
\[
\sum_{i=1}^{n} S_i P = \sum_{i=1}^{n} (z q_i + h x_i + a_i) P
= \sum_{i=1}^{n} (z q_i P + h x_i P + a_i P)
= \sum_{i=1}^{n} (z q_i P) + \sum_{i=1}^{n} (h x_i P) + \sum_{i=1}^{n} (a_i P)
= z \sum_{i=1}^{n} (Y_i + L_i) + h \sum_{i=1}^{n} X_i + \sum_{i=1}^{n} A_i
= z (Y_0 + L_0) + h X_0 + A_0 \tag{3}
\]
Hence the theorem is proved.

Theorem 2: The proposed LDMS scheme is linkable.

Proof: On receiving \( S_i \) and \( S_j (1 \leq i, j \leq n, i \neq j) \), \( ID_1 \) checks if the equation (2) holds.

Since we have
\[
S_i P - S_j P = (z q_i + h x_i + a_i) P - (z q_j + h x_j + a_j) P
= z (Q_i - Q_j) + h (X_i - X_j) + A_i - A_j
= z (Y_i - Y_j) + z (l_i K_i - l_j K_j)
+ h (X_i - X_j) + A_i - A_j \tag{4}
\]
If the equation (2) holds, it shows that the following equation holds,
\[
l_i K_i = l_j K_j \tag{5}
\]
where \( l_i = H_0 (K_i, ID_i) \), \( l_j = H_0 (K_j, ID_j) \). It is known that the equation (5) holds if and only if \( K_i = K_j \), \( ID_i = ID_j \), which means that \( S_i \) and \( S_j \), the two individual signatures are generated from the same malicious user, and the malicious user forges the signature by choosing two different one-time addresses to generate a multi-signature in a short time. Thus, \( ID_1 \) can link \( S_i \) and \( S_j \) together and reject them. If all of the individual signatures are valid, \( ID_1 \) computes \( S_0 = \sum_{i=1}^{n} S_i \) and outputs the final multi-signature \( (A_0, S_0, L_0) \) on \( m \).

Theorem 3: Under the adaptive chosen message attacks (CMA), if \( A \) can forge a valid signature of the proposed LDMS scheme in the random oracle model, then we can construct a probability polynomial time (PPT) algorithm to solve the ECDLP.

Proof: The security proof is based on the security model of [18]. The following game will be executed between an adversary \( A \) and a challenger \( C \). In order to prove security of the LDMS scheme under the CMA, the adversary \( A \)
and the challenger C play this game interactively. At the end of the game, assume the adversary A generates a valid signature \( \sigma^* \) on a message \( m^* \) corresponding to the signers \( \{ID^*_1, ID^*_2, \ldots, ID^*_n\} \). If the following conditions are satisfied, then we can say that A wins the Game.

1) The signature \( \sigma^* \) is valid on the message \( m^* \) corresponding to the signers \( \{ID^*_1, ID^*_2, \ldots, ID^*_n\} \).

2) Let \( ID^*_j, j \in R \{1, 2, \ldots, n\} \) be the honest signer, at least one of the cosigners is honest and it has never been queried by the oracle One-Time-Private-Key-queries, Real-Private-Key-queries and Real-Public-Key-queries.

3) \( ID^*_j \) and \( X^*_j \) have never executed the LDMS-Sign query for the message \( m^* \).

Assume that the adversary A breaks our LDMS scheme, then we will prove that there is a polynomial time bounded challenger C can solve an instance of ECDLP. Assume \( \{aP, k_j(aP)\} \) is a random instance of ECDLP, where \( k_j \in R Z^*_q \), and \( a \) is chosen randomly from \( Z^*_q \) by C. The challenger C sets the honest user \( ID^*_j \)'s real public address as \( aP, k_j(aP) \) and C tries to output \( k_j \) from the random instance \( \{aP, k_j(aP)\} \). The challenger C works with the adversary A interactively by the following ways:

- **Setup.** C sets the security parameter \( k \), the challenger C sets the system’s parameter set as \( \omega = \{H_i(0 = 1, 2), F_q, E/F_q, G_q, P\} \). Then \( \omega \) will be sent to the adversary A by the challenger C. The three hash functions \( H_0, H_1 \) and \( H_2 \) will be treated as random oracles. To avoid data inconsistency and response queries quickly, the challenger C creates five different initial-empty lists \( L_0, L_1, L_2, L_{ID}, L_{RID} \). Lists \( L_i(i = 0, 1, 2) \) contains three different hash results during the process of \( H_i(i = 0, 1, 2) \) queries, list \( L_{ID} \) contains the one-time public/private key pairs and list \( L_{RID} \) contains the real public/private key pairs. We donate the tuples included in the five lists as \( (l_i, K_i, ID_i), (h_i, m_i, A_0i, X_0i, ID_i), (z_i, m_i, A_0i, Y_0i, ID_i), (sk_i, pk_i, ID_i), (k_i, K_i, ID_i) \) respectively. The adversary A interacts with the challenger C by asking the following queries:

- **\( H_i(i = 0, 1, 2) \) queries.** If the adversary A inquires about the value of \( H_i \) with an identity \( ID_i \), the challenger C consults the list \( L_i \) and returns the corresponding hash value suppose that it has been queried before. If not, the challenger C chooses a number \( l_i(h_i/z_i) \in R Z^*_q \) and sends it to the adversary A. Then C inserts the tuples \( (l_i, K_i, ID_i), (h_i, m_i, A_0i, X_0i, ID_i), (z_i, m_i, A_0i, Y_0i, ID_i) \) to three lists \( L_i(i = 0, 1, 2) \) respectively.

- **Real-Private/Public-Key-queries.** If the adversary A inquires about the real private/public keys of the signer \( ID_i \), the challenger C refuses to reply if \( ID_i = ID_j \) holds. Else, the challenger C consults the tuple \( (k_i, K_i, ID_i) \) in the list \( L_{RID} \) if this query has been asked before and outputs the real private/public keys \( (k_i, K_i) \) to the adversary A. Otherwise, C chooses the number \( k_i \in R Z^*_q \), computes \( K_i = k_i(aP) \), and outputs the real private/public keys \( (k_i, K_i) \) to the adversary A and inserts the tuple \( (k_i, K_i, ID_i) \) in the list \( L_{RID} \).

- **One-Time-Private/Public-Key queries.** If the adversary A inquires about the one-time private/public keys of the signer \( ID_i \), the challenger C responds under different conditions. If this query has been asked before, the challenger C consults the tuple \( (sk_i, pk_i, ID_i) \) in the list \( L_{ID} \) and outputs the one-time private/public key \( (\bot, pk_i) \) to the adversary A if \( ID_i = ID_j \) holds, else, C outputs the one-time private/public key \( (sk_i, pk_i) \) to the adversary A. If this query has never been asked before, C searches the tuple \( (k_i, K_i, ID_i) \) in the list \( L_{RID} \). For the real public key \( K_i \), if the tuple is empty, C chooses the numbers \( q_i, k_i, l, x_i \in R Z^*_q \), computes \( Q_i = q_i(aP), X_i = x_i(aP), K_i = k_i(aP) \), sets \( H_0(K_i, ID_i) \) as \( l \), \( Y_i = Q_i \), \( l \), \( K_i, x_i = (x_i, q_i) \) and \( pk_i \) as \( (X_i, Y_i) \). Then C returns \( (\bot, pk_i) \) to the adversary A if \( ID_i = ID_j \) holds, else, C outputs \( (sk_i, pk_i) \) to the adversary A. Finally, C inserts the tuples \( (l_i, K_i, ID_i), (sk_i, pk_i, ID_i), (k_i, K_i, ID_i) \) in the lists \( L_0, L_{ID}, L_{RID} \) respectively.

- **Signing queries.** If the adversary A asks for the signature on a message \( m \) of the signer \( ID_i \), the challenger C generates the signature under different conditions. If \( ID_i = ID_j \) holds, the challenger C picks random numbers \( S_i, h_i, z_i \in R Z^*_q \) and \( l_i \in G_q \), computes \( A_i = -z_i(Y_i + L_i) - h_iX_i + S_i(aP) \) and returns \( (A_i, S_i, L_i) \) as the signature to the adversary A. Else, the challenger C searches the tuple \( (k_i, K_i, ID_i) \) in the list \( L_{RID} \) for the real public key \( K_i \), if the tuple is empty, C chooses \( a_i, l, h_i, z_i, k_i \in R Z^*_q \), sets \( H_0(K_i, ID_i) \) as \( l \), computes \( A_i = a_i(aP), K_i = k_i(aP), S_i = z_iq_i + h_iX_i + a_i, L_i = lK_i, \) and returns \( (A_i, S_i, L_i) \) as the signature to the adversary A. After collecting all of the individual signatures \( (A_i, S_i, L_i)(1 \leq i \leq n) \), the final multi-signature \( (A_0, S_0, L_0) \) can be computed by the adversary A.

- **Challenge phase.** The adversary A outputs a signature \( (A_0, S_0, L_0) \) with a hash value \( z \) corresponding to the signer \( ID_i(1 \leq i \leq n) \) on a chosen message \( m \). Suppose that there exists an honest signer \( ID_i \) who has never been asked by the One-Time-Private-Key queries, Real-Private-Key queries and Real-Public-Key queries. According to the forking lemma [18], the challenger C is demanded to generate another valid signature \( (A_0, S_0^*, L_0) \) on the same message \( m \) with a different hash value \( z^* \) if the above queries are repeated. According to two forged signatures \( (A_0, S_0, L_0) \) and \( (A_0, S_0^*, L_0) \), if both of them can be verified successfully, then we can write

\[
S_0(aP) = z(Y_0 + L_0) + hX_0 + A_0 \quad (6)
\]

\[
S_0^*(aP) = z^*(Y_0 + L_0) + hX_0 + A_0 \quad (7)
\]
From the equation (6) and (7), we get
\[ S_0'(aP) - S_0(aP) = z^* (Y_0 + L_0) + hX_0 + A_0 - z (Y_0 + L_0) - hX_0 - A_0 \]
\[ = (z - z^*) \sum_{i=1}^{n} (Y_i + L_i) \]
\[ = (z - z^*) \left( Y_j + L_j + \sum_{i=1, i \neq j}^{n} (Y_i + L_i) \right) \]
\[ = (z - z^*) \left( Y_j + \sum_{i=1, i \neq j}^{n} Q_i \right) \]
\[ = (z - z^*) \left( y_j aP + \sum_{i=1, i \neq j}^{n} q_i(aP) \right) \]
\[ = (z - z^*) \left( y_j + \sum_{i=1, i \neq j}^{n} q_i \right) (aP) + (z - z^*) \sum_{i=1, i \neq j}^{n} q_i(aP) \] (8)

From the equation (8), we can obtain
\[ (S_0^* - S_0)(aP) = (z - z^*)(y_j + \sum_{i=1, i \neq j}^{n} q_i) (aP) \] (9)

From the equation (9), we can obtain
\[ (S_0^* - S_0)/(z - z^*) = y_j + \sum_{i=1, i \neq j}^{n} q_i \] (10)

From the equation (10), we can obtain
\[ k_j = ((S_0^* - S_0)/(z - z^*)) - (y_j + \sum_{i=1, i \neq j}^{n} q_i))/l_j \] (11)

Thus, \( C \) solves a random instance \( \{ aP, k_j(aP) \} \) of ECDLP that contradicts the complexity assumption of ECDLP.

In this game, it is supposed that the adversary \( A \) knows the one-time and real private keys of all signers except the honest signer \( ID_j \), and the adversary \( A \) only tries to breach the proposed LDMS scheme against the honest signer \( ID_j \). Suppose that the adversary \( A \) does not execute the queries for more than one time for the same input and asks for at most \( q_H \) times \( H(i = 0, 1, 2) \) queries, \( q_{ok} \) times One-Time-Private-Key queries and \( q_1 \) times LDMS-Sign queries. According to the work of [19], the probability that the adversary solves the ECDLP is \( \varepsilon' \geq \frac{1}{n} \left( 1 - \frac{q_H}{q_{ok}} \right) \left( 1 - m q_1/m \right) \varepsilon \), where \( \varepsilon \) is the probability of breaking the security of the proposed LDMS scheme.

**IV. DELETING THE BLOCKCHAIN**

After proposing a new multi-signature scheme, this section will introduce a deletable blockchain in detail. We donate the transaction as \( tx = (ID, d) \), where \( ID \) is a unique transaction ID number for each transaction, \( d \) is the concealed transaction contents, the real transaction contents are donated as \( d' \). The consensus mechanism used in our blockchain is proof-of-space. In this section, we first describe how to propose a deletion request, and then give the details of the deletion protocol.

**A. PROPOSING A DELETION REQUEST**

In our proposed scheme, suppose that all of the transactions in a block are from the same transaction sender and we specify that only the sender of the transactions has the right to propose a deletion request. In an anonymous environment, in order to satisfy full transparency and accountability when deleting a block, we utilize different privacy-protected strategies according to different deletion reasons.

In Figure 3, we show a brief introduction to the process of proposing a deletion request. If a sender wishes to delete a block, he first classifies the deletion reasons. For the reason of an invalid identity, such as a failing company, the transaction sender can choose to reveal his worthless transaction identity. For the reason of expired transaction contents, such as abolished legal provisions, the transaction sender can choose to disclose the transaction contents. The details of the revelation process are shown as follows.

1) REVEALING THE SENDER’S REAL IDENTITY

If the transaction sender chooses to reveal his real identity, he needs to generate two traceable ring signatures [15] on the same tag \( L \), where \( L \) is a tag used in the signing process. We donate the \( L \) as \( (ID, pk_N) \), where \( ID \) is the unique transaction ID number, \( pk_N \) is the ring members in the traceable ring signature.

The specific process of revealing the identities is as follows:

1) As shown in Figure 4, each transaction sender needs to generate a traceable ring signature \( \sigma' \) on \( (L, m') \) when the transaction is generated, where \( m \) is the hash value of the transaction contents \( d \). We donate the signature \( \sigma' \) as \( (A_1, c_N, z_N) \). The sender sets \( \sigma' \) as a part of the transaction output and broadcasts \( \sigma' \) to the network.

2) If the sender chooses to reveal his real identity, the transaction sender needs to generate another traceable ring signature \( \sigma'' \) on \( (L, m') \), where \( m' \) is another hash value of the transaction contents \( d \). We donate the signature \( \sigma'' \) as \( (A_1, c_N, z_N) \). Then the sender broadcasts the deletion request \( r = (i, \sigma') \) to the network, where \( i \) is the block’s index number.
FIGURE 4. The process of revealing the identities.

FIGURE 5. The process of revealing the transactions.

3) The rest of the users verify two signatures \( \sigma, \sigma' \). The next step will be executed if and only if two signatures are both valid.

4) To extract the real signer who generates the two signature \( \sigma \) and \( \sigma' \) on \( m \) and \( m' \) with respect to the same tag \( L \), the rest of the users do the following steps:

a) Parse \( L \) as \((ID, pk_N)\). Compute \( h = H(L) \) and \( A_0 = H'(L, m) \), where \( H() \) and \( H'() \) are two distinct hash functions used in the traceable ring signature. For all \( i \in N \), compute \( \sigma_i = A_0 + hA_i \in G_q \). Repeat the same steps to retrieve \( \sigma_i' \) from the other signature \( \sigma' \), for all \( i \in N \).

b) For all \( i \in N \), if \( \sigma_i = \sigma_i' \) holds, store the corresponding public key \( pk_i \) in the tuple \( TraceList \).

c) Output \( pk \) if it is the only element in \( TraceList \), and the revealed identity is \( pk \).

2) REVEALING THE TRANSACTION CONTENTS

If the transaction sender chooses to disclose the real transaction contents, it needs to make a hash-based Pedersen commitment on the real transaction contents \( d' \).

The process of revealing the transactions contents is shown as follows:

1) As shown in Figure 5, before the deletion operation, the transaction sender picks two random numbers \( r_1 \) and \( r_2 \) and computes the commitment \( C = h(r_1, r_2, d') \), where \( h \) is a one-way hash function. Then the sender sets the commitment \( C \) and the random number \( r_1 \) as a part of the transaction output but keeps \( d' \) and \( r_2 \) secret. Thus, we can extend the transaction as \( tx = (ID, d, \sigma, C, r_1) \).

2) If the sender chooses to disclose his real transaction contents \( d' \) as the deletion evidence, it sets the deletion request as \( r = (i, d', r_2) \) and broadcasts \( r \) to the network, where \( i \) is the index number of the block.

3) The rest of the users calculate the hash value \( h(r_1, r_2, d') \) and compare the hash result with the commitment \( C \) received in step 1 to verify the validity of the real transaction contents \( d' \). If the equation (12) holds, the revealed transaction contents \( d' \) is valid.

\[
h(r_1, r_2, d') = C \quad (12)
\]

B. THE DELETION PROTOCOL DESCRIPTION

In this sub-section, we give the details of the deletion protocol. A deletion policy is introduced to determine whether a deletion operation should be accepted or not, and a valid deleted block is said to satisfy the policy, i.e., \( P(Chain, B^*) = 1 \), if the number of the multi-signature signers who come to an agreement to delete the block \( B^* \) in the chain \( Chain \) is at least \( 2/3 \cdot N \), and \( N \) is the number of the users in the chain. A block is deleted by the following steps:

1) On receiving a deletion request \( r_i \) from the network, the rest of the users broadcast their vote results to the network according to the disclosed identity or transaction contents. If a user agrees on the deletion request, he sets his vote result as \( VoteDel = 1 \), else, he sets his vote result as \( VoteDel = 0 \).

2) The user who has given a vote on the request \( r_i \) collects all the vote results \( VoteDel = 1 \) with respect to the same deletion request \( r_i \) from the network. If the value of the last vote results is larger than \( 2/3 \cdot N \), all of the voters \( U \) who support the deletion request generate a linkable multi-signature \( LDMSig_{\text{ID}} = LDMSig_U (i, D_i, \zeta_i) \) on the deletion message \( m \) by using their one-time addresses, where \( m \) contains the following information:

a) The index of the block \( i \).

b) The deletion information \( D_i \) including the deletion time and the invalid identity of the transaction sender or the Pedersen commitment \( C \).

c) The miner’s signature \( \zeta_i \) on the transaction sub-block \( r_{i+1} \).

Then these users replace the original transaction contents \( d \) with the multi-signature \( LDMSig_{\text{ID}} \), create a deleted block \( B^*_i \) and broadcast the block \( B^*_i \) to the network.

3) On receiving \( B^*_i \) from the network, each user does the following steps to delete the corresponding block in their local chain:

a) Compares the signature \( \zeta_i \) included in the deletion message \( m \) with the signature \( \zeta_i \) included in the signature sub-block, checks if the multi-signature \( LDMSig_{\text{ID}} \) is valid. He continues the following steps if the two signatures are the same and the multi-signature is valid, else, the user aborts the deletion operation.

b) Computes the policy \( P(Chain, B^*_i) \) to check whether \( B^*_i \) should be accepted in the local chain.

c) Replaces \( B_i \) with \( B^*_i \) in the local chain if \( P(Chain, B^*_i) = 1 \) holds, else, the user aborts the deletion operation.
The structure of the $i$th block after the deletion operation is shown in Figure 6. It can be seen from Figure 6 that after the deletion operation on the $i$th block, the hash relationship and the signature relationship between the previous block and the latter one are unchanged, and the transaction sub-block is replaced by the multi-signature LDMSig. The rest of the block structure has never been changed. In this way, a large number of transactions are successfully released.

Our privacy-protected deletable blockchain protocol offers public verifications. To validate a deleted chain, the users first check each block exactly like in the underlying immutable blockchain. If an “empty” block is found, then the users check if the multi-signature is valid and check if the deleted block satisfies the deleting policy $P(\text{Chain}, B^*)$. Only if all of the verifications are successful, the deleted chain can be accepted.

C. SECURITY ANALYSIS

In this sub-section, we analyze security of the proposed privacy-protected deletable blockchain.

1) ONLY THE TRANSACTION SENDER CAN PROPOSE A DELETION REQUEST

In our proposed deletable blockchain, we suppose that all of the transactions in a block are from the same transaction sender. Due to the anonymity of the transaction contents and the addresses of both parties in one transaction, anyone except the transaction sender cannot see the real transaction contents or the real identities of both parties, the unrelated users cannot determine whether a transaction can be deleted or not. Therefore, only the sender of the transactions can reveal his real identity or disclose the real transaction contents, thus, only the sender has the absolute right to propose a deletion request and the rest of the users have no right to propose a deletion request.

2) THE PROCESS OF REVEALING THE IDENTITIES

The real transaction contents and the identity of the receiver are still concealed if the transaction sender chooses to reveal his real identity. Even if the identity of the transaction sender is exposed to the network, the receiver’s identity and the transaction contents are still private to everyone. In this way, our scheme can ensure that the user’s private information is protected to the greatest extent.

3) THE PROCESS OF REVEALING THE TRANSACTIONS

The identities of the transaction sender and the receiver are still concealed if the transaction sender chooses to disclose the real transaction contents. Even if the real transaction contents are exposed to the network, the identities of the transaction receiver and sender cannot be known to anybody. What’s more, the secure and efficient Petersen commitment scheme used in sub-section A satisfies the following two security properties:

- **Message hiding property.** The hash value $h(\bar{r}_1, \bar{r}_2, d')$ and the random number $r_1$ are the commitment evidences of the transaction sender to the whole network. In the first step, the transaction sender uses the hash function and the random numbers to prevent any malicious user in the blockchain from inverting the function to obtain the concealed message (the real transaction contents $d'$), that is, the one-way hash function ensures the concealment.

- **Message binding property.** The collision-free performance of hash function ensures that the transaction sender cannot find another data $d''$ and a random number $r_2' \neq r_2$ that satisfy the equation $h(\bar{r}_1, \bar{r}_2, d') = h(\bar{r}_1, \bar{r}', d'')$. In this way, the transaction sender cannot deceive anybody, and everyone can ensure that the transaction sender has indeed revealed the real transaction contents $d'$.

4) TWO INVALID SUB-SIGNATURES GENERATED BY THE SAME USER CAN BE LINKED

In our privacy-protected blockchain system, the users use the one-time addresses to generate a linkable multi-signature during the deleting process. They use the one-time addresses as their pseudonyms to protect their identity privacy. As shown in equation (5), if a malicious user generates two sub-signatures by using two different one-time addresses to satisfy the deletion policy $P(\text{Chain}, B^*)$ in a short time, the designated collector $ID_1$ can link the two invalid sub-signatures and rejects them.

V. EXPERIMENTS

We conduct several experiments to evaluate the efficiency of our proposed privacy-protected blockchain system. We invoke the functions in TomMath library [20] to manipulate big integer numbers. SHA-256 and DSA-512 are used when the blockchain is generated and ECC-200 is used to implement elliptic curve groups. The programs are written in C++ language with the following specifications:

- 8GB of RAM
- Intel(R) Core (TM) i7-5500U CPU @ 2.40GHz
- Windows 10
- Visual Studio Ultimate 2013
A. MINING PROCESS UNDER THE PROOF-OF-SPACE CONSSENSUS MECHANISM

We simulate the mining process under the proof-of-space consensus mechanism. Suppose that the system generates a directed acyclic graph containing a total of 30 nodes. The generated graph is shown in Figure 7.

We suppose that there are 7 miners in the system donated as A, B, C, D, E, F, and G respectively. Each miner can calculate the label value of each node in the directed graph (using SHA-256) according to his own public key. Each miner generates a directed graph with a fixed structure, and there is an one-to-one correlation between the label value of each node and the miner’s identity.

Due to the limitation of the storage space, when the system picks some nodes randomly and publishes a challenge, the miner with larger storage space can always return the corresponding label values quicker and obtains the accounting rights more easily. The public keys of the seven miners are shown as follows:

\[ PK_A = 286F550A8EE6AC6..., \]
\[ PK_B = 7C3141B6A24FA9C..., \]
\[ PK_C = 178940D61F1A1FD0..., \]
\[ PK_D = 2B7761A25A8BCDA..., \]
\[ PK_E = 138FEF1EA5FE340A..., \]
\[ PK_F = 104B64D762388874B..., \]
\[ PK_G = 2786196EEE846AD6.... \]

Assume that A, B, C, D, E and F stores 12, 15, 18, 21, 24 and 27 node label values respectively, and miner G stores 30 node label values which means that G can store the complete directed graph. Each miner plays the role of a space certifier \( P \) to prove that he has enough space to store the whole directed graph to the verifier \( V \), who is played by the system. The verifier \( V \) publishes a challenge which is denoted as \( C = \{ \psi_{c1}, \psi_{c2}, ..., \psi_{ck} \} \), then he initiates the challenge and asks \( P \) to return the corresponding label values and their parent nodes’ label values to check whether \( P \) has enough space to store the complete directed graph. Then \( P \) returns the corresponding label values as the response to \( V \).

In Table 1, we record the time consumption of responses to different challenges with respect to seven miners.

It can be seen from Table 1 that under different challenges \( C \), it is faster for \( P \) to respond to different challenges if \( P \) possesses larger storage space, and it is easier for \( P \) to obtain the accounting rights if \( P \) returns the answers more quickly. This is the basic mining process under the proof-of-space consensus mechanism.

B. THE GENERATION OF THE NEW BLOCK

The miner who obtains the right of accounting has the right to generate a new block. As shown in Figure 8, we take block 35 as an example to introduce how to generate a proof sub-block, a signature sub-block, and a transaction sub-block respectively.

The proof sub-block records the hash value \( \psi_{35} = \text{Hash}(i, \psi_i, (p_k, \gamma_i, c_i, a_i)) \), where \( i = 35 \) is the block’s index number, \( \psi_i = "13B0344FA38ADA831616A30EC A41D51BE6C98B0" \) is the DSA signature on the signature sub-block \( \psi_{34} \), \( (p_k, \gamma_i, c_i, a_i) \) is the proof information under the proof-of-space consensus mechanism, \( p_k \) is the public address of the miner, \( \gamma_i = "3B8777C7A6D1B291F464951DF19 CE3AB38E184FD7951126B4B2A60591D52723E6" \) is the hash proof of the directed acyclic graph structure generated by a Merkle tree algorithm, \( c_i \) is the challenge received from the verifier \( V \) during the mining process, and \( a_i \) is the response to the challenge generated by the miner.

We can obtain the hash result of the 35th proof sub-block by computing the hash value of all the data mentioned above and the hash value is:

| Challenge | A | B | C | D | E | F | G |
|-----------|---|---|---|---|---|---|---|
| Time(ms)  | 172ms | 1783ms | 4055ms | 7287ms | 165ms | 999ms | 2777ms | 6342ms | 169ms | 308ms | 1693ms | 4036ms | 176ms | 281ms | 1092ms | 2934ms | 163ms | 287ms | 517ms | 1695ms | 170ms | 277ms | 442ms | 1088ms | 169ms | 276ms | 285ms | 287ms |

### TABLE 1. Time consumption of responses to different challenges.
\[ \varphi_{35} \text{=} "294B1584F49003885EAE7A5E742652B089A656193D676FE2FDCC7D9DF0F78F1B6". \]

The signature sub-block \( \sigma_i = \{ i, \zeta_i, \zeta_0 \} \) records the index number \( i \) and two signatures \( \zeta_i \) and \( \zeta_0 \), where \( \zeta_i = "2AE9566BCB22A8F5582204B38EC528DBA58E78C5A0" \) is the DSA signature on the transaction sub-block \( t_35 \) of the 35th block, and \( \zeta_0 \) is the DSA signature on the signature sub-block of the 34th block.

The transaction sub-block \( t_i = \{ i, ctx \} \) records the index number \( i \) and 100 transactions \( tx \).

### C. Deleting Transactions

Assume that the transactions of the 34th block are overdue, in order to save the storage space, the sender of the transactions reveals his own identity, generates a traceable ring signature \( \sigma' \) and broadcasts the deletion request \( r = (i, \sigma') \) to the network.

The rest of the users check the relationship between the two traceable ring signatures \( (\sigma, \sigma') \) and trace the sender's real public address \( pk \), the revealed public address is shown as follows:

\[
pk = 62FE1DF891F6C9EBB8675520483953209615B0536C0D816424320260F7A2A6E757CF78EC52A058BE8650B5AD8F03D5A19.
\]

Assume that the rest of the users in the blockchain system come to a deletion agreement on October 2, 2019. According to the deletion reason \( pk \), the deletion time \( "20191002" \), the index number \( i = 34 \), and the signature \( \zeta_i = "750A7DFB41A85C41DA313F14F90F85F1AF92F425" \) of the 34th transaction sub-block, they generate a deletion message \( m \), where \( m \) is shown as follows:

\[
m = 62FE1DF891F6C9EBB8675520483953209615B0536C0D816424320260F7A2A6E757CF78EC52A058BE8650B5AD8F03D5A192019100234750A7DFB41A85C41DA313F14F90F85F1AF92F425.
\]

Finally, these users generate an LDMS multi-signature \((A_0, S_0, L_0)\) on the deletion message \( m \). The results of the LDMS multi-signature is:

\[
A_0 = ED34368C12E34D4DDE4B44AF98A09DEDF840A2094F8A5C01.
\]

\[
S_0 = (128FBBCC712147ACDE190768DC09B33395BF12F2360FB9AD157, 8AD17ED6AA72F14E4380D231D1CB5840BD770E2ECF8C43199).
\]

\[
L_0 = (2E01CD93E7C14F93AC0395AAA925A09DEF0F2360FB9AD157, 8AD17ED6AA72F14E4380D231D1CB5840BD770E2ECF8C43199).
\]

After the deletion operation, the deletion performers broadcast a new block \( B^* \) that contains \((A_0, S_0, L_0)\) in the transaction sub-block instead of the original 100 transactions to the network. All the users in the network verify the multi-signature \((A_0, S_0, L_0)\), compare the signature \( \zeta_i \) included in the deletion message \( m \) with the signature \( \zeta_i \) included in the signature sub-block and calculate the deletion policy \( P(\text{Chain}, B^*) \). If all of the verifications are successful, the verifier replaces the original block with the new block \( B^* \) in their local chain. The new block structure is shown in Figure 9.

### REFERENCES

[1] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://bitcoin.org/bitcoin.pdf
XIANJAI CAI received the B.S. degree in electronics and information engineering from Shanghai University, China, in 2018, where she is currently pursuing the M.S. degree. Her research interests are blockchain security and cryptocurrencies.

YANLI REN received the M.S. degree in applied mathematics from Shaanxi Normal University, China, in 2005, and the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University, China, in 2009. She is currently a Professor with the School of Communication and Information Engineering, Shanghai University, China. She has published more than 60 articles on international journals and conference papers. Her research interests include applied cryptography, secure outsourcing computing, and network security.

XINPENG ZHANG received the B.S. degree in computational mathematics from Jilin University, China, in 1995, and the M.S. and Ph.D. degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. He was with the State University of New York at Binghamton, as a Visiting Scholar, from January 2010 to January 2011, and Konstanz University, China, in 2018, where she is currently pursuing the M.S. degree. Her research interests include blockchain security and cryptocurrencies.

[2] I. Eyal, A. E. Gencer, and E. G. Sirer, “Bitcoin-NG: A scalable blockchain protocol,” in Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI), Santa Clara, CA, USA, 2016, pp. 45–59.

[3] H. Cho, “ASIC–resistance of multi-hash proof-of-work mechanisms for blockchain consensus protocols,” IEEE Access, vol. 6, pp. 66210–66222, 2018.

[4] C. Badertscher, P. Gaži, and A. Kiayias, “Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. Toronto, ON, Canada: Allstream Centre, 2018, pp. 913–930.

[5] S. Park, A. Kwon, G. Fuchsbaier, P. Gaži, J. Alwen, and K. Pietrzak, “SpaceMint: A cryptocurrency based on proofs of space,” in Proc. Int. Conf. Financial Cryptogr. Data Secur., Berlin, Germany, 2018, pp. 480–499.

[6] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “ZeroCash: Decentralized anonymous payments from bitcoin,” in Proc. IEEE Symp. Secur. Privacy, Berkeley, CA, USA, May 2014, pp. 459–474.

[7] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, “RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero,” in Proc. Eur. Symp. Res. Comput. Secur., Cham, Switzerland, 2017, pp. 456–474.

[8] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, “Mixcoin: Anonymity for bitcoin with accountable mixes,” in Proc. Int. Conf. Financial Cryptogr. Data Secur., Berlin, Germany, 2014, pp. 486–504.

[9] J. Camenisch, D. Derler, S. Krenn, and H. C. Pöhl, “Chameleon-hashes with ephemeral trapdoors,” in Proc. IACR Int. Workshop Public Key Cryptogr., Berlin, Germany, 2017, pp. 152–182.

[10] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, “Redactable blockchain—Or—Rewriting history in bitcoin and friends,” in Proc. IEEE Eur. Symp. Secur. Privacy, Paris, France, Apr. 2017, pp. 111–126.

[11] K. Ashiritha, M. Sindhu, and K. V. Lakshmy, “Redactable blockchain using enhanced chameleon hash function,” in Proc. 5th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS), Tamil Nadu, India, 2019, pp. 323–328.

[12] D. Derler, K. Samelin, D. Slamanig, and C. Striecks, “Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based,” in Proc. 26th Netw. Distrib. Syst. Secur. Symp., San Diego, CA, USA, 2019.

[13] Y. Ren, D. Xu, X. Zhang, and D. Gu, “Deletable blockchain based on threshold ring signature,” J. Communications, vol. 40, no. 04, pp. 71–82, Apr. 2019, doi: 10.11959/j.issn.1000-436x.2019084.

[14] S. Noether, A. Mackenzie, and T. M. Research Lab, “Ring confidential transactions,” Ledger, vol. 1, pp. 1–18, Dec. 2016.

[15] E. Fujisaki and K. Suzuki, “Traceable ring signature,” in Proc. Int. Workshop Public Key Cryptogr., Berlin, Germany, 2007, pp. 181–200.

[16] N. Van Saberhagen, “CryptoNote v 2.0,” Tech. Rep., 2013.

[17] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in Proc. 1st ACM Conf. Comput. Commun. Secur., 1993, pp. 62–73.

[18] M. Bellare and G. Neven, “Multi-signatures in the plain public-key model and a general forking lemma,” in Proc. 13th ACM Conf. Comput. Commun. Secur., Songdo, South Korea, 2006, pp. 390–399.

[19] S. Hafizul Islam, M. Sabzinejad Farash, G. Biswas, M. Khurram Khan, and M. S. Obaidat, “A pairing-free certificateless digital multisignature scheme using elliptic curve cryptography,” Int. J. Comput. Math., vol. 94, no. 1, pp. 39–55, Jan. 2017.

[20] Torn. St. Denis. LibTomMath. Accessed: Nov. 25, 2019. [Online]. Available: https://sourceforge.net/projects/tommath/