Research on Efficiency Optimization Methods for Blockchain-Based Identity Authentication Mechanisms: A Postprint

Authors: Lingtao Tang, Xu Min, Jin Yurong

Date: 2018-06-19T00:00:00+00:00

Abstract

Blockchain-based decentralized PKI replaces traditional Certificate Authorities (CAs) with blockchain, leveraging its authority and transparency to achieve decentralized authentication. Most existing research employs blockchain traversal methods to query identity-public key pairs for verifying whether a public key belongs to a communication counterpart, which suffers from low efficiency. This paper proposes an identity authentication scheme based on cryptographic accumulators, which maps on-chain identity and public key information into accumulator values. This approach enhances the verification efficiency of identity-public key pairs while fulfilling authentication functionality, simultaneously addressing the insufficient storage problem of light nodes amid continuous blockchain growth, and experimentally validates the feasibility and effectiveness of the proposed method.

Full Text

Preamble

Article URL: http://www.arocmag.com/article/02-2019-10-020.html
Journal: ChinaXiv Partner Journal - *Computer Application Research*
Title: Research on Methods for Improving Efficiency of Identity Authentication Based on Blockchain

Authors: Tang Lingtao, Xu Min, Jin Yurong
Affiliation: Jiangnan Institute of Computing Technology, Wuxi, Jiangsu 214083, China

Abstract: Decentralized public key infrastructure (PKI) based on blockchain replaces traditional certificate authorities (CAs) with blockchain, leveraging its

authority and transparency to achieve decentralized authentication. Current research primarily verifies whether a public key belongs to a communication counterpart by traversing the blockchain to query identity-public key pairs, which is inefficient. This paper proposes an identity authentication method based on cryptographic accumulators that maps on-chain identity and public key information into an accumulated value, thereby improving verification efficiency while addressing the insufficient storage space of lightweight nodes as the blockchain continuously grows. Experiments demonstrate the feasibility and effectiveness of the proposed method.

Keywords: blockchain; cryptographic accumulator; decentralization; identity authentication

0 Introduction

Public key infrastructure (PKI) is a collection of hardware, software, and security policies that provides a complete security mechanism, enabling users to communicate and conduct e-commerce transactions based on certificates through a series of trust relationships, even when they are unaware of each other' s identities or geographically distributed. As the foundation and core of current network security construction, PKI is the fundamental guarantee for secure e-commerce. A traditional PKI system includes certificate authorities (CAs), PKI policies, hardware/software systems, registration authorities (RAs), certificate distribution systems, and PKI applications. However, CA-centric PKI suffers from single-point failure and prominent intrusion targets, which has spurred the development of decentralized PKI.

Blockchain technology has attracted significant attention since Satoshi Nakamoto proposed Bitcoin in 2008. As an immutable public ledger, blockchain has been widely used in online payments, distributed storage, and anti-counterfeiting certification. One implementation of decentralized PKI utilizes blockchain to store identity and public key information. During authentication, the current approach traverses the blockchain to query certificates, checks whether the public key belongs to the claimed identity, and finally sends challenge messages to verify digital signatures. However, as an append-only public chain, blockchain' s data volume continuously grows. In recent years, blockchain size has exceeded 100 GB and will continue to increase, making traversal-based methods increasingly inefficient. If traditional methods persist, authentication time will fail to meet practical requirements. Meanwhile, devices such as mobile phones cannot store such massive data. The introduction of cryptographic accumulators can solve the efficiency problem in member verification during identity authentication.

Cryptographic accumulator schemes contain algorithms that map a set of multiple elements into a single accumulated value and provide a compact witness to prove that a given value indeed belongs to the set. Starting from this context,

this paper proposes an efficiency optimization method for identity authentication in decentralized PKI using cryptographic accumulators. The study focuses on the definition, implementation principles, and characteristic comparisons of cryptographic accumulators, as well as their application in member verification combined with blockchain to improve authentication efficiency.

1.1 One-Way Accumulators

The concept of cryptographic accumulators originated from one-way accumulators first proposed by Benaloh et al. One-way accumulators are defined as a family of one-way hash functions with quasi-commutativity. Unlike common hash functions with a single argument, these consider two input arguments.

Definition 1 (One-Way Hash Functions). A family of one-way hash functions is an infinite sequence of functions in set $\mathcal{H}$, where $h_\lambda \in \mathcal{H}$ (with λ as a security parameter), satisfying:

a) For any integer λ and any input, computation is polynomial-time feasible.

b) For any polynomial-time algorithm $\mathcal{A}$, the probability satisfies certain negligible conditions.

These functions are computable and one-way: given h_λ, computing $h_\lambda(x, y)$ is polynomial-time feasible, while finding collisions is computationally infeasible.

Definition 2 (Quasi-Commutativity). A function $f : X \times Y \to X$ is quasi-commutative if:

$$f(f(x, y_1), y_2) = f(f(x, y_2), y_1)$$

for all $x \in X$ and $y_1, y_2 \in Y$.

Quasi-commutativity ensures that for a set of values, the accumulated hash value remains unchanged regardless of the computation order when the initial value is fixed. A one-way accumulator is a one-way function with quasi-commutativity that maps a set Y to a constant value z and provides proofs for each member, enabling member verification.

1.2 Strong One-Way Accumulators

Definition 1 shows that finding collisions is negligible when variables are properly selected, but does not consider attackers who might forge credentials by improperly selecting values. To enhance security, strong one-wayness is introduced.

Definition 3 (Strong One-Way Hash Functions). A family of strong one-way hash functions is an infinite sequence of functions in set $\mathcal{H}$, where $h_\lambda \in \mathcal{H}$ (with k as a security parameter), satisfying:

a) For any integer k and any input, computation is polynomial-time feasible.

b) For any polynomial-time algorithm $\mathcal{A}$, the probability of finding collisions is negligible.

The probability depends on random choices of the algorithm and random output. Strong one-way accumulators prevent attackers from constructing fake member values to forge witnesses.

1.3 Collision-Free Accumulators

Baric et al. proposed that cryptographic accumulators require stricter properties. Under strong one-wayness, attackers might still carefully forge member values to construct witnesses. Therefore, collision-free accumulators are introduced.

Definition 4 (Accumulator Scheme). A cryptographic accumulator scheme is a quadruple of polynomial-time algorithms $(\text{Gen}, \text{Eval}, \text{Wit}, \text{Ver})$:

- **Gen**: A probabilistic algorithm generating initial parameters. It takes a security parameter λ and accumulator threshold N (the maximum number of values that can be securely accumulated) and returns an accumulator key k.
- **Eval**: A probabilistic algorithm computing the accumulated value of all values in set Y. It takes k and Y and outputs an accumulated value z and auxiliary information aux.
- **Wit**: A probabilistic witness generation algorithm. It takes k, a value $y_i \in Y$, and auxiliary information aux; if y_i is indeed in Y, it outputs a witness w_i proving y_i was accumulated.
- **Ver**: A deterministic verification algorithm. It takes k, y_i, witness w_i, and accumulated value z to verify whether y_i was accumulated, outputting Yes or No.

Definition 5 (Collision-Freeness). An accumulator scheme is N-collision-free if for any integer N and any polynomial-time probabilistic algorithm $\mathcal{A}$, the probability of forging a witness is negligible. If an accumulator is N-collision-free for any positive integer N, it is collision-free. Collision-free accumulators prevent attackers from forging witnesses by constructing false member sets.

1.4 Dynamic Accumulators

Applications in member authentication require accumulators to support efficient updates to accumulated values and member witnesses when the member set changes, particularly for efficient member revocation. Otherwise, all members would need to recompute accumulated values and witnesses whenever members are added or removed, making accumulators inefficient for practical applications.

Definition 6 (Dynamic Accumulator Scheme). A dynamic accumulator scheme is a heptuple of polynomial-time algorithms (Gen, Eval, Wit, Ver, Add, Del, Upd):

- **Gen, Eval, Wit, Ver**: Same as in Definition 4, forming the basic structure.
- **Add**: Element addition algorithm (usually deterministic). Takes accumulator key k, accumulated value z of set Y (with $|Y| < N$), and value y' to add; returns new accumulated value z' and update information aux_{add}.
- **Del**: Element deletion algorithm (usually deterministic). Takes accumulator key k, accumulated value z of set Y, and value y to delete; returns new accumulated value z' and update information aux_{del}.
- **Upd**: Witness update algorithm (deterministic). Updates witness w_i for existing elements after Add or Del operations. Takes k, w_i, y_i, operation $\text{op} \in \{\text{Add}, \text{Del}\}$, and update information; returns updated witness w_i'.

1.5 Features and Comparison

Key characteristics of cryptographic accumulators serve as evaluation criteria:

- **Dynamicity**: Efficient element addition/deletion and witness update algorithms (see Definition 6).
- **Robustness**: Accumulator administrators need not be trusted; trapdoor information cannot forge witnesses.
- **Universality**: Accumulators provide both membership and non-membership witnesses.
- **Security Assumption**: Member verification functionality remains secure against attackers under stated assumptions.
- **Compactness**: Accumulators map large sets to smaller-order accumulated values, with small storage requirements for accumulated values and witnesses, and low time complexity for update algorithms.

Performance characteristics of various accumulators are shown in Table 1 (where n is the number of elements in the set).

Table 1. Characteristics of Various Cryptographic Accumulators

Accumulator	Dynamicity	Robustness	Universality	Witness/Accumulator Size	Security Assumption
BeMa[8]				$O(1)$	Strong RSA + Random Oracle
BarPif[11]				$O(1)$	Strong RSA
CamLys[12]				$O(1)$	Strong RSA
LLX[15]				$O(1)$	Pairings
AWSM[16]				$O(\log n)$	Collision-resistant Hash
CHKO[14]				$O(1)$	Strong RSA

2 Cryptographic Accumulator-Based Authentication Methods

2.1 Decentralized Member Verification

The quasi-commutativity of one-way accumulators ensures that with a fixed initial value x_0, the accumulated hash value z for values $y_1, y_2, ..., y_m$ remains unchanged regardless of computation order. That is, for any permutation π of the accumulated value z and values y_i, we have:

$$z = h(\cdots h(h(x_0, y_{\pi(1)}), y_{\pi(2)}), \cdots, y_{\pi(m)})$$

Additionally, the one-way property ensures that given z and y_i, finding a witness w_i for a non-member $y' \notin Y$ is negligible. Thus, accumulators enable member verification: given a set Y, all elements' accumulated hash value z can be computed. A member holding y_i can compute the accumulated value of all other elements and provide witness w_i to prove membership. The verifier can check if $z = h(w_i, y_i)$. Non-members attempting to forge identities must construct a witness w' for a fake value y', which is computationally infeasible under certain assumptions.

This approach does not hide user information since all y_i are used to compute z, but it significantly reduces storage overhead by eliminating the need to maintain member lists. Unlike traditional centralized authorities maintaining member lists, accumulators enable mutual identity verification without central participation.

2.2 Designing Cryptographic Accumulator-Based Identity-Public Key Pair Authentication

Traditional identity verification between parties A and B proceeds as follows: a) A sends (id_A, pk_A) to B, claiming public key pk_A belongs to identity id_A. b) B queries blockchain (or authoritative third party) for matching registration information (id_A, pk_A) to verify the public key' s ownership. c) B sends random challenge string ch to A, who digitally signs it: $\sigma = \text{Sign}_{sk_A}(ch)$. d) B verifies the signature using pk_A: if $\text{Ver}_{pk_A}(\sigma, ch) = 1$, it proves A holds private key sk_A and thus owns identity id_A.

In blockchain-based decentralized identity authentication, cryptographic accumulators play different roles depending on node status. For full nodes, querying all locally stored chain information requires traversing the entire blockchain, with authentication efficiency decreasing as blockchain volume grows. For lightweight nodes, there is insufficient space to store the complete blockchain, preventing normal communication authentication unless they request queries from full nodes, which again faces traversal inefficiency.

We improve steps a) and b) as follows: A sends (id_A, pk_A, w_A) to B, where w_A is the witness. B runs the verification algorithm from Definition 4:

$$\text{Ver}(k, (id_A, pk_A), w_A, z) \to \{0, 1\}$$

to determine whether (id_A, pk_A) belongs to accumulator z. This typically reduces the time complexity of step b) from $O(n)$ to $O(\log n)$ or $O(1)$. Using cryptographic accumulators, authentication functionality can be achieved by storing only the accumulated value and related information, reducing local storage space complexity from $O(n)$ to $O(\log n)$ or $O(1)$, enabling lightweight nodes to participate normally.

2.3 Accumulator Operation Methods on Blockchain

Integrating cryptographic accumulators with existing blockchain-based decentralized identity authentication improves authentication efficiency and addresses lightweight node storage limitations. Compared to conventional blockchains, we add a "current accumulated value" field to transaction information, with each participating node maintaining its own witness locally.

This section describes accumulator creation from the genesis block and its operation during identity registration, revocation, update, and verification.

2.3.1 Initial Accumulator Creation The miner node creates security parameters: $k \leftarrow \text{Gen}(1^\lambda, N)$. Through Eval, it inputs an initial member list of m elements (at least containing the miner itself) as (id, pk) pairs to obtain the initial accumulated value z_0, which is placed in the "genesis block"and broadcast to the network. All recipients verify the correctness of z_0 creation. If correct,

they synchronize the block and compute their own witnesses w_i; otherwise, they discard it. The algorithm is described in Algorithm 1.

Algorithm 1. Initial Accumulator Creation

```
for creator:
    k = Gen(1^ , N)
    initial_idlist = [id_1, id_2, ..., id_m]
    initial_pklist = [pk_1, pk_2, ..., pk_m]

    for i in range(len(initial_idlist)):
        current_transaction[i]['identity'] = initial_idlist[i]
        current_transaction[i]['publickey'] = initial_pklist[i]
        y_i = hash(id_i, pk_i)

    (z_0, aux) = Eval(k, {y_1, ..., y_m})
    current_transaction[i]['accu'] = z_0
    genesis_block.add(current_transaction)
    proof = PoW(genesis_block_header)
    genesis_block.add(proof)

    # Compute own witness (assuming index 1)
    w_1 = Wit(k, y_1, aux)
    blockchain.append(genesis_block)
    broadcast(genesis_block)

for neighbor_i in node.neighbors:
    when receive genesis_block from node:
        if current_block is constructed correctly:
            blockchain.append(genesis_block)
            # Update own witness based on new identities in genesis_block
            for (id, pk) in genesis_block:
                w_i = Wit(k, hash(id, pk), aux)
        else:
            abort genesis_block
```

2.3.2 Identity ID Registration When a miner approves a node' s identity registration request, it adds (id, pk) to the accumulator, computes the new accumulated value z_{new} and corresponding witness w_{id}. The miner then includes (id, pk, z_{new}) in a newly mined block.

All block recipients verify the correctness of z_{new}. If valid, they update their locally stored accumulated value and update their witnesses via Upd; otherwise, they discard the block. The algorithm is described in Algorithm 2.

Algorithm 2. Identity Registration

```
for miner in miners:
```

```
    when receive (id, pk, sig(sk_id, "register")):
        current_transaction = []

        if is_well_constructed(id, pk, sig):
            current_transaction.add((id, pk, "register"))

            # Generate new accumulated value and witness
            y = hash(id, pk)
            (z_new, aux) = Add(k, z_old, y)
            w_id = Wit(k, y, aux)
            current_transaction.add((z_new, w_id))

            current_block.add(current_transaction)
            proof = PoW(current_block_header)
            current_block.add(proof)

            # Update own witness (assuming index j)
            w_j = Upd(k, w_j, y, "Add", aux)
            blockchain.append(current_block)
            broadcast(current_block)

            for neighbor_i in node.neighbors:
                when receive current_block from node:
                    if current_block is constructed correctly:
                        blockchain.append(current_block)
                    else:
                        abort current_block
```

2.3.3 Identity ID Revocation The miner verifies that the identity exists in the accumulator. If verification succeeds, it removes the node' s identity, recomputes the accumulated value z_{new}, and includes it in a new block. If verification fails, the process aborts.

All block recipients verify z_{new} correctness. If valid, they update their locally stored accumulated value and witnesses via Upd; otherwise, they discard the block. The algorithm is described in Algorithm 3.

Algorithm 3. Identity Revocation

```
for miner in miners:
    when receive (id, pk, w, sig(sk_id, "revoke")):
        current_transaction = []

        if Ver(k, (id, pk), w, z) == 1 and Ver(sig, pk, "revoke") == 1:
            # Identity is registered and revocation command is authorized
            current_transaction.add((id, pk, "revoked"))
```

```
# Generate new accumulated value
y = hash(id, pk)
(z_new, aux) = Del(k, z_old, y)
current_transaction.add(z_new)

current_block.add(current_transaction)
proof = PoW(current_block_header)
current_block.add(proof)

# Update own witness (assuming index j)
w_j = Upd(k, w_j, y, "Del", aux)
blockchain.append(current_block)
broadcast(current_block)

for neighbor_i in node.neighbors:
    when receive current_block from node:
        if current_block is constructed correctly:
            blockchain.append(current_block)
        else:
            abort current_block
```

2.3.4 Public Key Update Public key update is equivalent to performing identity revocation followed by registration. The user presents witness w, the miner revokes (id, pk) via Algorithm 3, then the user provides new public key pk', and the miner registers (id, pk') via Algorithm 2.

2.3.5 Identity Verification Given identity-public key pair (id, pk) and witness w, any user can verify legitimacy by running $\text{Ver}(k, (id, pk), w, z)$.

3.1 Block Construction

This work adopts the strong RSA assumption-based cryptographic accumulator from [8,9], modifying block construction code based on blockchain generation methods. Partial code is shown below, with the resulting block structure illustrated in Figure 1 [Figure 1: see original paper].

```
def new_block(self, proof, pre_hash):
    block = {
        'index': len(self.chain) + 1,
        'timestamp': time.time(),
        'transactions': self.current_transactions,
        'proof': proof,
        'pre_hash': pre_hash or self.hash(self.chain[-1]),
    }
```

```
# Update accumulated value and witness
if len(self.chain) > 0:
    idpkbind = self.current_transactions[0]['identity'] + \
               self.current_transactions[0]['publickey']
    idpkhash = self.hash(idpkbind)
    idpkconvert = int(idpkhash, 16)

    for x in range(len(self.chain)):
        self.witness[x] = pow(self.witness[x], idpkconvert, self.n)

    self.witness.append(self.accu)
    self.accu = pow(self.accu, idpkconvert, self.n)

# Reset the current list of transactions
self.current_transactions = []
self.chain.append(block)
return block
```

3.2 Results and Analysis

Experiments were conducted on three PCs with Windows 7 64-bit, Intel® Xeon(R) CPU E7520 @ 1.86 GHz, and 16 GB DDR3 RAM.

Six datasets of identity-public key pairs were created, ranging from 5,000 to 30,000 entries in increments of 5,000. Identities consisted of 6-20 random digits and letters; public keys comprised 15 random digits and letters. The strong RSA accumulator threshold was a 150-digit decimal integer (product of two strong primes generated via Miller-Rabin primality test). The seed x was a random integer not exceeding N.

Figure 2 [Figure 2: see original paper] compares the time cost between traditional blockchain traversal and accumulator-based authentication. Traditional traversal time grows linearly with blockchain length, while strong RSA accumulator verification time is constant, independent of blockchain length, fluctuating only slightly due to machine conditions at different operation times.

However, accumulator updates and maintenance require additional computation. Two methods are considered:

a) Mining nodes maintain all member witnesses.

b) Full nodes only maintain the block structure in Figure 1. Mining nodes compute only their own witness; each node updates its witness independently based on block content. For offline users, they record current block height i and request synchronization from full nodes upon reconnection.

Figure 3 [Figure 3: see original paper] shows blockchain construction speed for both methods versus traditional blockchain. When miners maintain all node witnesses, time grows non-linearly—requiring $O(n)$ witness updates per block. Without computing other nodes' witnesses, each node performs only one accumulator evaluation per block, with time curves nearly overlapping traditional methods, demonstrating negligible impact on full node computation.

To ensure strong RSA accumulator security, experiments tested larger modulus sizes. Figure 4 [Figure 4: see original paper] shows that single-step authentication time grows non-linearly with modulus size but remains at millisecond level even beyond 1,024 bits (308 decimal digits), fully acceptable for ordinary users.

Using a 320-digit decimal modulus (meeting security requirements), blockchain construction experiments were repeated. As shown in Figure 5 [Figure 5: see original paper], Method 1 shows significantly increased construction time with larger modulus, while Method 2' s overhead is only millisecond-level per block, making construction time essentially unchanged.

These results demonstrate that introducing cryptographic accumulators with node-maintained witnesses improves authentication efficiency over traditional methods, with improvements increasing as blockchain grows. The additional computation for block construction is negligible compared to Proof-of-Work, and the system remains efficient even with security-enhanced large moduli.

---

4 Conclusion

In recent years, Bitcoin and various altcoins have gained increasing attention, mostly at the digital currency level. As an underlying technology, blockchain has few practical applications. This paper advances the frontier by studying and improving blockchain applications in decentralized PKI. It first analyzes cryptographic accumulator definitions, principles, and compares mainstream accumulators across different criteria. It then introduces accumulators to improve identity-public key authentication efficiency, reducing time and space complexity. Depending on the accumulator, traditional $O(n)$ verification complexity can be reduced to $O(\log n)$ or even $O(1)$. Experiments verify that strong RSA-based accumulators achieve constant-time identity-public key verification, demonstrating correctness and effectiveness, significantly improving decentralized PKI verification efficiency for large-scale blockchains.

Since existing strong RSA accumulators lack robustness and have limitations without trusted administrators (cannot guarantee trapdoor information won' t be abused for unauthorized user deletion), future work should adopt collision-resistant hash-based Merkle Tree-type accumulators for better robustness.

References

[1] Garman C, Green M, Miers I. Decentralized anonymous credentials [C]// Proc of Network and Distributed System Security Symposium. 2014.

[2] Kulynych B, Isaakidis M, Troncoso C, et al. ClaimChain: decentralized public key infrastructure [J]. arXiv preprint arXiv: 1707.06279, 2017.

[3] Anada H, Kawamoto J, Weng Jian, et al. Identity-embedding method for decentralized public-key infrastructure [C]// Proc of International Conference on Trusted Systems. Cham: Springer, 2014: 1-14.

[4] Morselli R, Bhattacharjee B, Katz J, et al. Keychains: a decentralized public-key infrastructure [R]. [S. l.]: University of Maryland, College Park College Park United States, 2006.

[5] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system [J]. Consulted, 2008.

[6] 袁勇, 王飞跃. 区块链技术发展现状与展望 [J]. 自动化学报, 2016, 42(4): 481-494. (Yuan Yong, Wang Feiyue. Blockchain: the state of the art and future trends [J]. ACTA automatica Sinica, 2016, 42(4): 481-494.)

[7] Miers I, Garman C, Green M, et al. Zerocoin: anonymous distributed e-cash from bitcoin [C]// Proc of IEEE Symposium on Security and Privacy. 2013: 397-411.

[8] Benaloh J, De Mare M. One-way accumulators: a decentralized alternative to digital signatures [C]// Proc of Workshop on the Theory and Application of Cryptographic Techniques. Berlin: Springer, 1993: 274-285.

[9] Nyberg K. Commutativity in cryptography [C]// Proc of the 1st International Workshop on Functional Analysis at Trier University. [S. l.]: Walter de Gruyter & Co, 1996: 331-342.

[10] Fazio N, Nicolosi A. Cryptographic accumulators: definitions, constructions and applications [J]. Paper written for course at New York University: www.cs.nyu.edu//nicolosi//papers//accumulators.pdf, 2002.

[11] Barić N, Pfitzmann B. Collision-free accumulators and fail-stop signature schemes without trees [C]// Proc of International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 1997: 480-494.

[12] Camenisch J, Lysyanskaya A. Dynamic accumulators and application to efficient revocation of anonymous credentials [C]// Proc of Annual International Cryptology Conference. Berlin: Springer, 2002: 61-76.

[13] 万国根, 周世杰, 秦志光. 单向累计函数技术分析 [J]. 计算机科学, 2005, 32(8): 57-59. (Wan Guogen, Zhou Shijie, Qin Zhiguang. An overview of the one-way accumulator technology [J]. Computer Science, 2005, 32(8): 57-59.)

[14] Camacho P, Hevia A, Kiwi M, et al. Strong accumulators from collision-resistant hashing [C]// Proc of International Conference on Information Security. Berlin: Springer, 2008: 471-486.

[15] Li Jiangtao, Li Ninghui, Xue Rui. Universal accumulators with efficient non-membership proofs [C]// Proc of Applied Cryptography and Network Security. Berlin: Springer, 2007: 253-269.

[16] Au M H, Wu Qianhong, Susilo W, et al. Compact e-cash from bounded accumulator [C]// Proc of Cryptographers'Track at the RSA Conference. Berlin: Springer, 2007: 178-195.

[17] 陈泽文, 王继林, 黄继武, 等. ACJT 群签名方案中成员撤消的高效实现 [J]. 软件学报, 2005, 16(1): 151-157. (Chen Zewen, Wang Jilin, Huang Jiwu, et al. An efficient revocation algorithm in ACJT group signature [J]. Journal of Software, 2005, 17(1): 33-50.)

[18] 邵奇峰, 金澈清, 张召, 等. 区块链技术: 架构及进展 [J]. 计算机学报, 2017: 1-20. (Shao Qifeng, Jin Cheqing, Zhang Zhao, et al. Blockchain technology: architecture and progress [J]. Chinese Journal of Computers, 2017: 1-20.)

[19] Sasson E B, Chiesa A, Garman C, et al. Zerocash: decentralized anonymous payments from bitcoin [C]// Proc of IEEE Symposium on Security and Privacy. 2014: 459-474.

Note: Figure translations are in progress. See original paper for figures.