Hindawi
Security and Communication Networks
Volume 2022, Article ID 9946088, 19 pages
https://doi.org/10.1155/2022/9946088

Research Article

Privacy-Oriented Transaction for Public Blockchain via Secret Sharing

Bo Mi[1], **Bingqing Wu**[1], **Darong Huang**[1], **Yang Liu**[1], **Lu Chen**[2] and **Shaohua Wan**[3,4]

[1]*School of Information Science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China*
[2]*Department of Information Security, Naval University of Engineering, Wuhan 430032, China*
[3]*Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China*
[4]*Department of Information Systems, King Abdulaziz University, Jeddah 21589, Saudi Arabia*

Correspondence should be addressed to Lu Chen; ieucl@163.com

Received 16 March 2022; Revised 13 May 2022; Accepted 16 July 2022; Published 19 November 2022

Academic Editor: Mukesh Soni

Immutability and traceability are the main reasons for the popularity of blockchain. Nevertheless, its transparency also makes the transactions visible to all participants, which seriously violates the privacy of some dealers. In order to transfer accounts over blockchain, all verifiers should be empowered with the ability to confirm the transactions, leading to the conflict between extensive consensus and individual privacy. Orienting to privacy issues of UTXO (Unspent Transaction Output), this paper exploited the unconditional security of Shamir's secret sharing to construct a logic-physical map for public chain, which enabled noninteractive transaction verification without privacy infringement. A comprehensive analysis indicated that the proposed scheme is secure under UC (Universal Composability) framework with practical computation and communication overheads.

1. Introduction

As the most popular distributed platform, blockchain is provided with the merits of traceability and immutability, attracting more and more users to adopt it for accounting and trading [1–3]. However, the transparency of blockchain makes records visible to all participants, even for those in consortium and private blockchain, which brings about severe threats to sensitive information such as business secrets or personal privacy. Nevertheless, the property of openness is inevitable for blockchain due to tampering-resistant and verification purposes. Therefore, the conflict between the availability and confidentiality of blockchain has become a significant obstacle to its extensive application.

The first showcase of privacy infringement for blockchain was presented in literature [4], taking Bitcoin as an illustrative example. It is pointed out that, due to the open nature of blockchain, attackers can statically analyse the distributed database and actively monitor the network flow, thus revealing the trading graph or transaction pattern of the Bitcoin system. Thereafter, Bitcoin is no longer anonymous as the inventor claims, not mentioning the confidentiality of ledgers. Accordingly, literature [4] also suggested two rudimentary tools that can be used to preserve the privacy of blockchain, namely, NIZK (Noninteractive Zero-Knowledge Proof) and HE (Homomorphic Encryption). Zero-Knowledge Proof allows the prover to convince the verifier that a certain assertion is correct without providing any useful information. Noninteractive Zero-Knowledge Proof does not require an interactive process and only needs to provide proof to the verifier to prove that he is correct, avoiding the possibility of collusion, but additional machines and procedures may be required to determine the order of experiments. In the blockchain, addresses are used to represent the parties to a transaction, thereby achieving anonymity. However, although the information on the chain is anonymous, the information in the real world is bound by the information on the chain. For example, many transactions are bound with the address on the chain and the bank account in the real world, which makes it convenient to trace

the transaction parties in the real world and weakens anonymity. Homomorphic Encryption allows ciphertext to be processed directly, and the result is still ciphertext. That is, direct processing of ciphertext is the same as processing and then encrypting the plaintext. Noninteractive Zero-Knowledge Proofs and Homomorphic Encryption can ensure a valid transaction while hiding other details such as sender, receiver, and transaction amount. However, while NIZK eliminates the need for multiple interactions, its biggest disadvantage is that the generation of proof is very slow. Zcash [5], for example, takes 30 seconds, while the popular Filecoin [6] takes an hour. Similarly, for homomorphic computation, its computational efficiency is also a factor that seriously restricts the block generation speed. By introducing extra security mechanisms into the blockchain, the overall impact of privacy threats can be further reduced with remarkable efficiency and reliability improvements. Since Critical Energy Infrastructure (CEI) systems are vulnerable to cyberattack and data privacy leakage when deep learning training, Lu [7] designed the fairness-aware and time-sensitive task allocation mechanisms in asynchronous federated learning for CEI.

The most intuitive way to preserve privacy is to upload encrypted information to the blockchain that no node can decrypt except the one who holds the secret key. However, this solution contradicts the basic principle of blockchain to some extent since it is a distributed database whose blocks must be commonly approved and maintained. Therefore, an open problem in such a research direction is how to make ciphertexts verifiable and computable, especially when they are encrypted by distinct keys. A promising way to conceal sensitive data in blockchain draws support from Attribute-Based Encryption. Since the keys are not directly related to each user but only their attributes, the participants provided with similar properties can hierarchically form a group. That is to say, they can share and process their collective data together, while any specific identity is concealed. Howbeit, considering that once a group member is corrupted, all of his keys would be exposed to the attacker. That is to say, such a solution is not fair enough, even if just a passive attack occurs.

Considering that blockchain is essentially a distributed data structure, many scholars have studied the privacy issues in distributed systems [8–11]. Wu et al. [12] propose an efficient identity-based equal test encryption scheme with bilinear pairings to solve the problem of efficiently searching encrypted data outsourced to the cloud. For distributed secure computing, they also propose an efficient and secure searchable encryption protocol using trap door substitution functions that can be deployed over the cloud-based Internet of Things [13]. Due to the decentralized nature of blockchain, some researchers attempt to draw support from secret sharing (SS) for privacy-preserving. Zyskind and Nathan [14] implement a protocol that transforms the blockchain into an automatic access control manager that does not require third-party trust. In [15], it says the best way to protect private data is never to let the service see the raw data but to allow it to run calculations directly on the network and get the final result. If we split the data into shares rather than encryption, we can securely evaluate any function using secure multiparty computation. Raman and Varshney [16] combine distributed storage, private key encryption, and Shamir's secret sharing scheme to distribute transaction data to ensure data integrity. In this scheme, hash and Merkle root are divided into secret shares, and each node only holds the corresponding share, which means each node only holds a part of the chain but can recover the whole chain, thus reducing the storage cost. Bartolucci et al. [17] construct a secret shared voting protocol based on blockchain; they use Shamir's secret sharing for on-chain vote submission and winning candidate determination. The dealer divides the private keys of the two candidates into secret shares and distributes them. Only when a sufficient number of shares were collected could the private keys of the two candidates be recovered and the winner of the voting be determined at the same time. And the protocol is combined with a shuffling technique to delink voters from their submissions. Such innovation is inspired by the fact that sensitive information can be separately allocated to distinct participants in the blockchain. Thus, it cannot be revealed or processed unless multiparties collaborate. Since the basic principle of secret sharing is similar to the consensus protocol in distributed systems, it is more natural and feasible to be combined with blockchain. Following this research direction, Zheng et al. [18] proposed a credible data sharing scheme, which takes advantage of the consistency and availability of blockchain to prevent shared data from tampering or modification. It exploits the Paillier cipher system to achieve the confidentiality of shared data. Specifically, this scheme stores encrypted data in blockchain and divides secret keys as distributed shares for security purposes. However, it requires a trusted third party to distribute the secret keys, which is an inevitable bottleneck with respect to both efficiency and security. Moreover, due to the limitations of the Paillier algorithm, it only realizes the operation of homomorphic addition rather than multiplication and Boolean calculations. To eliminate the necessity of a key server, Kim et al. [19] assembled Shamir's secret sharing scheme, private key encryption, and information diffusion algorithm to construct a distributed secret sharing blockchain. By secretly sharing private keys and hash values on diverse nodes, the purpose of cooperative decryption is achieved. Nevertheless, how to directly calculate between encrypted data remain unaddressed, which severely hinders the practicality of their scheme. Similarly, Fukumitsu et al. [20] presented a secure online storage system that divides user data into parts and distributes them over a P2P network through anonymous communication. Though achieved better performance, the data stored in the blockchain are fragments containing more or less semantic information of their primitives. Moreover, this scheme is still incapable of processing the encrypted data, which introduces more challenges to authentication, consensus, and operation of blockchain.

To not only narrow the gap between security and maneuverability but also fit the distributed structure of blockchain, a practical privacy-preserving scheme will be presented in this paper. We exploit Shamir's secret sharing (SSS) to retain sensitive data on a logic blockchain which is physically composed of multiple share-piece chains. That is to say, the data are stored in the form of secret shares to

ensure their confidentiality. Thanks to the Q2 structure of Shamir's secret sharing, the secret cannot be recovered unless at least t pieces of shares are collected, where t stands for a predefined threshold. To make sure that the shares can be efficiently processed, we also presented a noninteractive homomorphic multiplication method followed by nonlinear operations such as logic calculation and comparison. Thereafter, transactions can be carried out without exposing any sensitive information within in chain. In this way, blockchain will not just be simple cloud storage but a secure platform with homomorphic function.

Briefly, the contributions of this paper are as the following:

(1) We proposed a homomorphic multiplication method based on Shamir's secret sharing, which eliminated the bottleneck of interactions among participants. Based on it, the concealed data can be processed in terms of nonlinear functions without communication delay.

(2) Combining SSS with the raft algorithm, a secure data computation model is presented, which not only ensures the confidentiality of data on blockchain but also guarantees that the computational result is recoverable when necessary. It means that the basic functions of blockchain can be normally and securely executed, including consensus, audit, and transaction.

(3) By actualizing basic operations such as unbounded fan-in addition and numerical comparison, a complete framework for secure UTXO (Unspent Transaction Outputs) is presented as an instance of our SSS-based blockchain scheme. Within such a framework, the mechanism of UTXO can be accurately and efficiently realized without infringement. We can also conclude that our scheme is practical for more complex blockchain since it is flexible and naturally consistent with decentralized systems.

2. Noninteractive Multiplication Based on Shamir's Secret Sharing

2.1. Preliminaries. We assume that a distributed network has n parties $P_1, P_2, P_3 \ldots P_n$ connected, and the index i of each party P_i is public. We continue to use the symbolic representation in [21]. Let $[a]_p$ represent a polynomial sharing of a secret a as in [22]. That means a is shared by a random polynomial $f_a(x) = a + a_1x + a_2x + a_3x + \cdots a_{t-1}x^{t-1} \bmod p$ and a_i, $(1 \le i \le t-1, t-1 < n/2)$ is randomly chosen from Z_p. Among them, p is an odd prime, Z_p is a prime field, and $a \in Z_p$ donates $a \in \{0, \ldots p-1\}$. Each party can get a secret share $f_a(i)$ calculated by its own index number.

In this work, it involves a Boolean comparison operation. We use $[C]_p$ to represent a polynomial sharing of a Boolean operation result, and $C \in \{0, 1\}$ is a Boolean test. For example, when $a < b$ is true, then $[a < b]_p = [1]_p$.

Shamir's secret sharing scheme has some very convenient properties. When we write $[a]_p + [b]_p$, it means that the secret $a + b$ can be revealed by calculating $f_a(i) + f_b(i)$ independently of each party where a and b are both secrets. In the same way, each party can compute $c + f_a(i)$, $cf_a(i)$ without interaction. And $c + a$, ca can be revealed by gathering t parties' shares where c is a constant. Calculating the multiplication of the two secrets ab by $[a]_p \times [b]_p$ is more complicated. In [23], an interactive multiplication has been proposed, and it requires each party to communicate interactively, which would result in a severe increase in computational complexity and time. To solve this problem, this paper proposes a noninteractive multiplication in which nodes can be computed locally.

2.2. Interactive Multiplication. For $[a]_p \times [b]_p$, in simple terms, the original version of multiplication is to distribute the secret shares that have already been allocated by a new random polynomial. However, this will lead to a significant increase in secret shares, and the number of nodes involved in decryption will increase accordingly. Next, we will briefly introduce interactive multiplication.

For the following two random polynomials, where a and b are two secrets and the secret shares assigned to each node are represented by $[a]_p$ and $[b]_p$, we need to get from $[a]_p$ and $[b]_p$ to $[ab]_p$.

$$\begin{aligned} f_a(x) &= a + a_1x + a_2x^2 + \cdots + a_{t-1}x^{t-1}, \\ f_b(x) &= b + b_1x + b_2x^2 \cdots + b_{t-1}x^{t-1}. \end{aligned} \tag{1}$$

Each node gets $f_a(x_i)$ and $f_b(x_i)$, so $f_{ab}(x_i)$ is available:

$$f_{ab}(x) = ab + r_1x + r_2x^2 \cdots + r_{2t-2}x^{2t-2}. \tag{2}$$

By using the Lagrange interpolation formula, we know that

$$\begin{aligned} f_{ab}(0) &= \sum_{j=0}^{2t-2} f_{ab}(x_j) l_j(0) \\ &= ab, \end{aligned} \tag{3}$$

$$l_j(0) = \prod_{0 \le m \le 2t-m \ne j} \frac{-x_m}{x_j - x_m}.$$

Since the order is $2t-2$, it needs to be distributed again to reduce its order to $t-1$ so that shares can be computed directly. Then we need to construct $2t-1$ new random polynomials:

$$2t-1 \begin{cases} h_{x_1}(x) = f_{ab}(x_1) + a_1'x + a_2'x^2 + \cdots + a_{t-1}'x^{t-1}, \\ h_{x_2}(x) = f_{ab}(x_2) + a_1''x + a_2''x^2 + \cdots + a_{t-1}''x^{t-1}, \\ \vdots, \\ h_{x_{2t-1}}(x) = f_{ab}(x_{2t-1}) + a_1^{(2t-1)}x \\ + a_2^{(2t-1)}x^2 + \cdots + a_{t-1}^{(2t-1)}x^{t-1}. \end{cases} \tag{4}$$

For one of the nodes x_i,

$$\begin{aligned} h_{x_i}(x) &= f_{ab}(x_i) + a_1^i x + a_2^i x^2 + \cdots + a_{t-1}^i x^{t-1} \\ &= f_a(x_i) \cdot f_b(x_i) + a_1^i x + a_2^i x^2 + \cdots + a_{t-1}^i x^{t-1}. \end{aligned} \tag{5}$$

x_i uses the above polynomial to calculate the corresponding shares of other nodes and sends $h_{x_j}(x_j)$. Let

$$H(x) = \sum_{j=0}^{2t-2} l_j(0) h_{x_j}(x). \tag{6}$$

$$\begin{aligned} H(0) &= l_0(0) f_{ab}(x_0) + l_1(0) f_{ab}(x_1) \\ &\quad + \cdots + l_{2t-2}(0) f_{ab}(x_{2t}) \\ &= ab, \end{aligned} \tag{7}$$

where $H(x)$ is the shared random polynomial of ab. After each node receives all$h_{x_j}(x_j)$, calculate$H(j) = \sum_{x_i=0}^{2t-2} l_i(x) h_{x_i}(x_j)$and then obtain$[ab]_p$.

It can be seen from the above process that, in the interaction process, additional random polynomials $h_{x_i}(x)$ need to be constructed, and the corresponding secret shares will increase by $(2t-1)n$. However, in blockchain, this will significantly reduce the efficiency of consensus, making it unable to meet the needs of transactions. Therefore, we design a noninteractive multiplication to eliminate the unnecessary extra traffic.

2.3. Noninteractive Multiplication. Besides homomorphic addition, efficient multiplication is also needed in the trading calculation. However, the multiplication of secret shares mentioned in Section 2.2 essentially shares the product of two shares again as a new secret. And this requires interaction between nodes. To realize the goal that nodes can complete all secret shares locally, we designed a multiplication secret sharing without interaction.

The form of direct multiplication of two secret shares $f_a(x_i)$ and $f_b(x_i)$ is

$$f_a(x_i) f_b(x_i) = ab + r_1 x + r_2 x^2 + \cdots r_{2t-2} x_i^{2t-2}. \tag{8}$$

Convert polynomials into vector multiplication, and the two random polynomials are shown as follows:

$$f_a(x_i) = (a, a_1, a_2, \ldots, a_{t-1})\left(x_i^0, x_i^1, x_i^2, \ldots, x_1^{t-1}\right)^T. \tag{9}$$

And

$$f_b(x_i) = (b, b_1, b_2, \ldots, b_{t-1})\left(x_i^0, x_i^1, x_i^2, \ldots, x_1^{t-1}\right)^T. \tag{10}$$

Then $f_a(x_i) f_b(x_i) = \alpha + \beta$, where

$$\begin{aligned} \alpha &= \left(\left(ab, ab_1 + ba_1, \ldots, \sum_{\sigma+\tau=\mu} a_\sigma b_\tau, \ldots, \sum_{\sigma+\tau=t-1} a_\sigma b_\tau\right) \bmod p\right) \\ &\quad \cdot \left(x_i^0, x_i^1, \ldots x_i^\mu, \ldots, x_i^{t-1}\right)^T, \\ \beta &= \left(\left(\sum_{\sigma+\tau=\mu} a_\sigma b_\tau, \ldots, \sum_{\sigma+\tau=\omega} a_\sigma b_\tau, \ldots, a_{t-2} b_{t-1} + a_{t-1} b_{t-2}, a_{t-1} b_{t-1}\right) \bmod p\right) \\ &\quad \cdot \left(x_i^{t-1}\left(x_i^1, \ldots, x_i^{\omega=t+1}, \ldots, x_i^{t-2}, x_i^{t-1}\right)\right)^T. \end{aligned} \tag{11}$$

From the above formulas, we can see that the order of α is precisely the order $t-1$ we need, and its leading coefficient is ab. So the secret share of ab belonging to node i can be obtained by subtracting $f_a(x_i) f_b(x_i)$ from β, which is α.

Based on the above observations, first, construct a random polynomial as follows:

$$f_s(x) = s + a_1 x + a_2 x^2 + a_3 x^3 + \cdots a_{t-1} x^{t-1} \bmod \left(x_i^t - 1\right), \tag{12}$$

where s is secret and $f_s(x_i)$ is a secret share distributed to i, where $p < x_i < q$. Because the node i owns all the information of $f_a(x_i)$, $f_b(x_i)$, and x_i, node i can obtain the secret share of ab by calculating

$$\begin{aligned} f_{ab}(x_i) &= f_a(x_i) f_b(x_i) - \left(x_i^{-t} - 1\right)^{-1} f_a(x_i) f_b(x_i) \\ &\quad \bmod\left(x_i^t - 1\right) - f_a(x_i) f_b(x_i) \bmod \left(x_i^t - 1\right). \end{aligned} \tag{13}$$

Through the above operation, even if multiplication is needed in the transaction, the node can complete the secret share calculation locally without interacting.

2.3.1. Proof of Correctness. For the above scheme, we will prove its correctness:

$$\begin{aligned} f_a(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}, \\ f_b(x) &= b_0 + b_1 x + b_2 x^2 + \cdots + b_{t-1} x^{t-1}. \end{aligned} \tag{14}$$

And then

$$\begin{aligned} f_{ab}(x) &= f_a(x) \cdot f_b(x) \\ &= ab + c_1 x + c_2 x^2 + \cdots + c_{2t-2} x^{2t-2}, \\ f_{ab}(x) \bmod x^{t-1} &= ab + c_1 x + \cdots + c_{t-1} x^{t-1} \\ &\quad + t\left(c_t + \cdots + c_{t-2} x^{t-2}\right) \bmod x^{t-1}. \end{aligned} \tag{15}$$

And $x^t \bmod x^t - 1 = 1$. Let

$$\begin{aligned} ab + c_1 x + c_2 x^2 + \cdots c_{t-1} x^{t-1} &= k_1, \\ c_t + c_{t+1} x + \cdots + c_{2t-2} x^{t-2} &= k_2. \end{aligned} \tag{16}$$

The above formula can be converted to

$$f_{ab}(x)\bmod x^t - 1 = k_1 + k_2 \bmod x^t - 1$$
$$f_{ab}(x) = k_1 + x^t k_2. \tag{17}$$

And

$$f_{ab}(x)\bmod\left(x^t - 1 - f_{ab}(x)\right) = -\left(x^t - 1\right)\cdot k_2 \cdot \left(x^t - 1\right)^{-1} \cdot \left(f_{ab}(x)\bmod\left(x^t - 1 - f_{ab}(x)\right)\right) = -k_2. \tag{18}$$

So

$$\begin{aligned} & f_{ab}(x) + \left(x^t - 1\right)^{-1}\left(f_{ab}(x)\bmod x^t - 1 - f_{ab}(x)\right)\bmod\left(x^t - 1\right) \\ &= \left(k_1 + k_2 x^t - k_2\right)\bmod\left(x^t - 1\right) \\ &= k_1 + \left(x^t - 1\right)k_2 \bmod\left(x^t - 1\right) \\ &= k_1 \\ &= ab + c_1 x + c_2 x^2 + \cdots c_{t-1} x^{t-1}. \end{aligned} \tag{19}$$

2.3.2. Proof of Security. Next, we will conduct a simple analysis of its security.

Lemma 1. *As long as no more than t participants collude, the Shamir(t, n) threshold secret sharing scheme is unconditionally secure.*

Obviously, in the noninteractive multiplication scheme proposed in this paper, when participants multiply the shares $f_a(x_i), f_b(x_i)$ of two secret shares, there is no need for information interaction between participants. Therefore, according to Lemma 1, the noninteractive multiplication based on secret sharing proposed in this paper is unconditionally secure.

3. The Secure Data Comparison Model

3.1. Raft Protocol. As the core of the blockchain system, the consensus mechanism helps to maintain the consistency of nodes. Various mainstream consensus protocols are analysed in [24]. By measuring the performance of these consensus algorithms and combining with the characteristics required by our scheme, we decided to adopt the raft consensus protocol. The raft protocol can elect a temporary leader to make the system consists in a term. In a nutshell, the raft protocol breaks the consistency problem into three relatively independent subproblems: leader election, log replication, and safety.

Since the blockchain in this paper is a logic-physical mapping structure, there are some differences in the logical and physical consensus process, among which the logical consensus process is as follows. Figure 1 shows the consensus process on a logical chain.

3.1.1. Leader Election. Nodes in the system have three states: follower, candidate, and leader. All nodes are started as followers, and their terms are initialized to 0. When the election starts, all nodes set the timeout at the same time. When a timeout occurs, the node becomes a candidate and sends the vote requests to the other nodes immediately. The timeout node only votes for itself, and its term automatically increases by 1. If it gets a majority of the votes in the system, it will become the leader.

Since each node has a random timeout between 100 and 500 milliseconds, all the followers cannot become candidates and send the vote requests at the same time. In other words, the node that becomes a candidate first and sends the vote requests has the "first-mover advantage" of becoming the leader. But if two nodes happen to be candidates at the same time and split votes or the candidate does not receive more than half of the votes, then they will reset the timer and restart the election.

3.1.2. Log Replication. When there is a leader, the system enters the work period. The client sends all the requests to the leader, and the leader schedules the order of concurrent requests to ensure the consistency of its state with the followers. Raft does this by informing followers of these requests and the order in which they are executed. The leader and followers execute these requests in the same order to ensure a consistent state.

First, when a request is received, the leader writes it to the local log as an entry. Note that the entry's status is uncommitted at this time, and the leader will not update the local data, so it will not be readable. When a leader is in the system, the leader will send a heartbeat to the followers at regular intervals. These heartbeats serve two purposes. First, they inform followers that the leader is online. Second, they can be used to send entry. Since the leader sending heartbeats is periodic, in one period, the leader may receive more than one entry from the client. Leader records the entry sent by the client to the local log and constantly AppendEntries until the next period, and these entries will be sent to the followers in parallel.

When an AppendEntries is sent, the leader will add the new log entry's index position (prevLogIndex) and Leader Term number (TERM) in it. The nodes verify consistency through the index inside the entries. If most nodes write it to the local log and return success, the entry written in the leader's log will be marked as committed. When the leader responds to the client, the result will be informed to the followers with the next heartbeat. Upon receiving the heartbeat, the followers will also mark the entry as committed. This means that more than half of the nodes in the system are consistent.

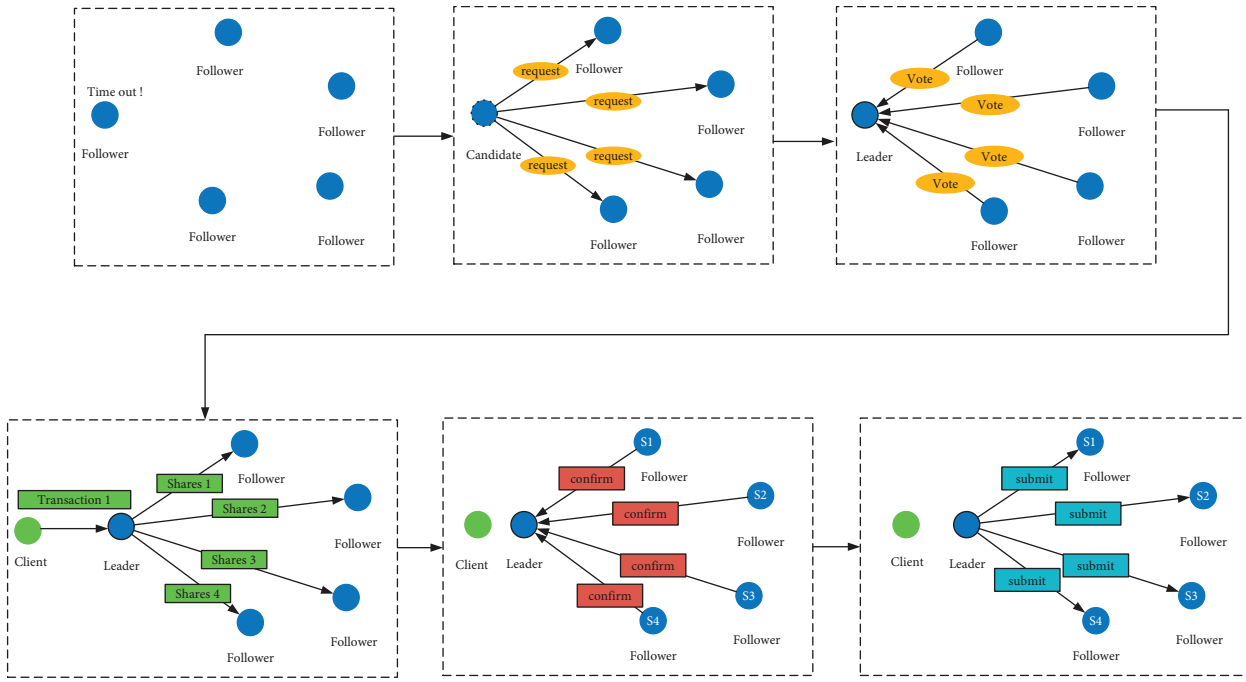

FIGURE 1: The consensus process on a logical chain. Ultimately, each node writes different secret shares to the local log.

If the follower does not find the same index position and term number in its log, it will refuse to receive the new entry. To solve the problem that the follower's logs are not consistent with the leader. The leader must find the position where the last log was not consistent, delete all the log entries from that position onwards, and send its logs to the follower.

3.1.3. Safety. However, the log replication phase does not sufficiently guarantee that each follower will execute the same instructions in the same order. For example, a follower may be unavailable while the leader has sent some log entries and when the follower is online and the leader is unavailable. The problem will arise if the follower is elected as the leader and others' log entries are rolled back.

To solve the above problem, the leader must store all the log entries that have been committed. The candidate must contact most nodes to win an election. This means that every log entry that has been committed must exist on at least one node. If the candidate's log is as new as most of the nodes, it must hold all the log entries that have been committed. Therefore, the raft protocol requires that the request must contain the candidate's log, and the voting nodes will reject the requests whose logs are not fresh enough. Determine whose logs are newer by comparing the term and the index of the last log in the two candidates' logs. If the last entry of the two logs has a different term, then the logs with a larger term number are newer. If the last entry in both logs has the same term, then the one with the longer log is newer.

If a leader crashes before committing a log entry, the subsequent leader will continue to try to duplicate the log. However, a leader cannot determine that a log entry from a previous term that is saved to the major of followers has been committed. This is obvious from the process of log replication. The raft protocol does not commit a log entry from a previous term by counting the number of replicas. Only the log entries for the current term of the leader can be committed by counting the number of replicas. Once the current term log entry is committed in this way, previous log entries will also be committed indirectly due to the log matching feature.

In fact, the public chain operates depending on the different physical chains stored by each node. As described above, in the early stage, each node votes for a leader node according to the leader election. Then, the leader will call the Comparison Protocol and Equality Test Protocol described in Sections 3.2 and 3.3 to determine whether the transaction meets the requirements. If so, the leader will inform each node to update data. The entire election and consensus process for the physical chain is shown in Figure 2.

Different from the consensus process on the logical chain, each follower receives a different secret share, and the leader is responsible for calling the protocol for verification. Only when the leader runs the protocols to send message 1 to the followers will each follower perform the calculation and update on the local chain.

3.2. Comparison Protocol. In the process of transaction verification, it is necessary to verify whether the payer's balance is enough to pay, which requires a comparison operation based on secret sharing. To implement the Comparison Protocol, Bitwise Sharing and LSB Protocol using wraparound should be implemented first. The specific call relationship is shown in Figure 3.

3.2.1. Bitwise Sharing. Bitwise Sharing is sharing a secret by sharing each bit. In a nutshell, each participant shares a secret $a \in Z_p$ in the form of $\{[a_{l-1}]_p, \ldots, [a_0]_p\}$, $a_i \in \{0, 1\}$, where $a = \sum_{i=0}^{l-1} 2^i a_i$. For simplicity of notation, use $[a]_B$ for Bitwise Sharing $\{[a_{l-1}]_p, \ldots, [a_0]_p\}$.

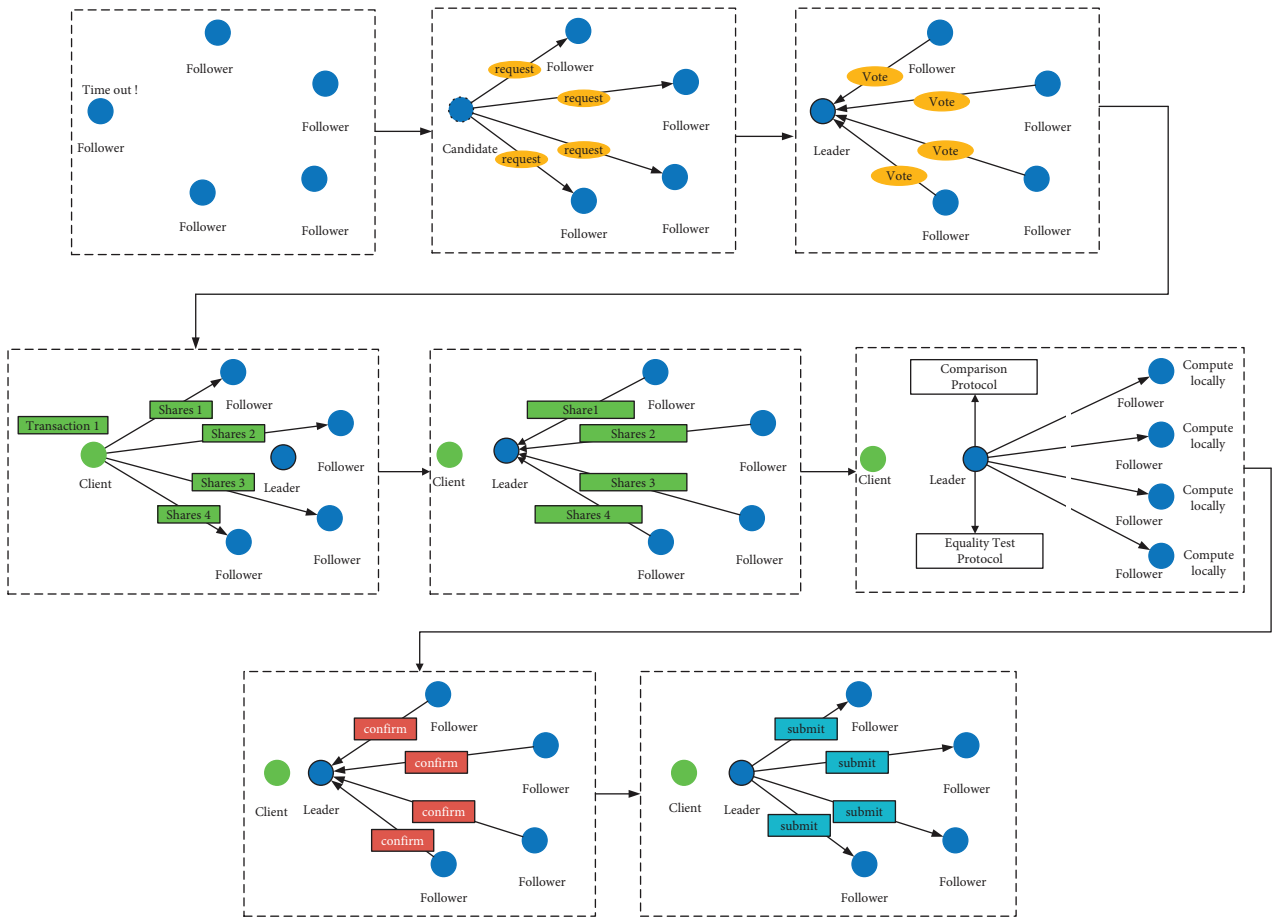

Figure 2: The consensus process on a physical chain. Each node computes on a local physical chain.

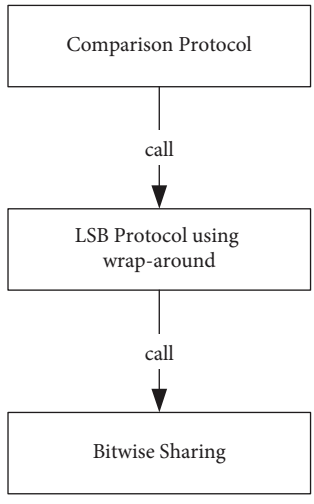

Figure 3: The call relationship of Comparison Protocol. Bitwise Sharing and LSB Protocol using wraparound are called subprotocols.

3.2.2. LSB Protocol Using Wraparound. To implement the Comparison Protocol, we first use the feature of wraparound under modulus p to estimate the size of the secret. We notice that, under the modulus p, the numbers less than p retain their properties, and the numbers greater than p wrap around. That means, if $a \in \{0, 1, 2 \ldots p - 1/2\}$, then $(2a \bmod p)_0 = 0$. If $a \in \{p - 1/2 + 1, \ldots, p - 1\}$, then $(2a \bmod p)_0 = 1$. Because when $a < p/2$, $2a \bmod p$ is even and when $a > p/2$, wraparound occurs, $2a \bmod p$ is odd. Thus, we can tell the magnitude of the secret a and $p/2$ by whether the lowest order after modulating p is 0 or 1.

We want to compute $[(a)_0]_p$ from $[a]_p$ without exposing a, and we just randomize the secret a to hide with a random number r and then calculate by bit with the random value obtained. Since we have r, we can get $[(a)_0]_p$ that we need. The whole calculation process is as follows:

Parties generate a Bitwise Sharing $[r]_B$ and get $[r]_p$.

Parties compute $[c]_p = [a]_p + [r]_p$ and reveal c.

If $a < p/2$ and no wraparound occurs, then $[(a)_0]_p = (c)_0 \oplus [(r)_0]_p$.

If $a > p/2$ and wraparound occurs, then $[(a)_0]_p = 1 - (c)_0 \oplus [(r)_0]_p$.

To see if there is a wraparound, we can compare r with c. If $r < c$, no wraparound occurs. And if $r > c$, wraparound occurs for $c = a + r - p$.

To sum up,

$$[(a)_0]_p = \left\{1 - \left([(c)_0 \oplus (r)_0]_p\right)\right\} \times [c <_B r]_p + \left\{(c)_0 \oplus [(r)_0]_p\right\} \times \left(1 - [c <_B r]_p\right), \tag{20}$$

where $[c <_B r]_p$ represents that c is less than r in bits, and the details are presented in [15]. In the actual calculation, the comparison result can be easily calculated by $[c <_B r]_p$, because all the parties that hold the bitwise shares of r and c are revealed. Therefore, we only need to calculate the result of $(c)_0 \oplus [(r)_0]_p$. Since the parties have revealed c, they can easily obtain $(c)_0$, and the value of $(c)_0$ can be used to determine the value of $(c)_0 \oplus [(r)_0]_p$. The specific calculation method is as follows:

$$(c)_0 \oplus [(r)_0]_p = \begin{cases} [(r)_0]_p \text{if } (c)_0 = 0, \\ 1 - [(r)_0]_p \text{if } (c)_0 = 1. \end{cases} \tag{21}$$

From the previous steps, we can obtain $[(a)_0]_p$ from $[(a)]_p$. And they can tell if a is greater than $p/2$ by $[(2a)_0]_p$, where $[a < p/2]_p = 1 - [(2a)_0]_p$.

Through the above protocol, we can effectively calculate the size relation between a and $p/2$ by using wraparound, laying a foundation for further comparison in the follow-up work.

3.2.3. Comparison. When Bitwise Sharing and LSB Protocol using wraparound are implemented in sequence, each participant can determine whether a secret a is less than $p/2$. Then, it can determine whether a is less than b through the different situations of the two secrets in the interval $[0, p/2]$. For the two secrets a and b that need to be compared, let $w = (a < p/2)$, $x = (b < p/2)$, $y = (a - b \bmod p < p/2)$, and $z = (a < b)$. The truth relation is shown in Table 1.

According to the truth table above, the expression for $z = (a < b)$ is

$$\begin{aligned} z &= w\bar{x} \vee \overline{wxy} \vee wx\bar{y} \\ &= w(1-x) + (1-w)(1-x)(1-y) + wx(1-y) \\ &= w(x + y - 2xy) + 1 - y - x + xy. \end{aligned} \tag{22}$$

3.3. Equality Test Protocol. When participants need to compare whether two secrets are equal, they need to implement Equality Test Protocol, which contains a relatively large number of subprotocols. Since we creatively propose noninteractive multiplication as the underlying module of Equality Test Protocol, the delay caused by asynchronous communication can be greatly reduced. In addition, as described in Figure 4, Unbounded Fan-In Or, Prefix-Or, Bitwise Less-Than, Joint Random Number Sharing, and so on are also called submodules.

Since there are many protocols nested in Equality Test Protocol, this section mainly describes the two protocols at the top of the call relationship. For details of other submodules, see [21]. Similarly, we give the symbolic representation of each subprotocol so as not to cause confusion when building subsequent protocols, as shown in Table 2.

Table 1: The truth case of different comparisons in the Comparison Protocol.

w	x	y	z
1	0	*	1
0	1	*	0
0	0	0	1
0	0	1	0
1	1	0	1
1	1	1	0

3.3.1. Joint Random Number Bitwise Sharing. To generate a shared random number, according to Bitwise Sharing, participants need to generate Bitwise Sharing for each bit of r_i and calculate $0 \le r = \sum_{i=0}^{l-1} 2^i r_i < p$. Then, participants generate l bits $[r_i \in_R \{0,1\}]_p$, $0 \le r \le l-1$ according to the Joint Random Bit Sharing. Finally, participants calculate $[r <_B p]_p$ by Bitwise Less-Than and recover the comparison result of $(r < p)$. If the recovered comparison result is 1, the generation of bitwise shared random number succeeds. If the result is 0, the protocol is executed again.

3.3.2. Unbounded Fan-In And. In the subsequent Equality Test Protocol, the Unbounded Fan-In And is necessary, which computes $[\vee_{i=0}^{l-1} a_i]_p$, where $[a_{l-1}]_p, \ldots, [a_0]_p$ and $a_i \in \{0,1\}$. As in [25], Damgård et al. proposed a symmetric Boolean function which only depends on the number of 1's in its input. With this framework, it can be concretized into Unbounded Fan-In And.

Give $[a_{l-1}]_p, \ldots, [a_0]_p$, where $a_i \in \{0,1\}$. Assume $A = 1 + \sum_{i=0}^{i=l-1} a_i$, and parties compute $[A]_p = 1 + \sum_{i=0}^{i=l-1} [a_i]_p$; that means A is the number of 1 in a_i plus 1. Then, parties can construct a l-degree polynomial $f_l(x)$ by $f_l(1) = 0$, $f_l(2) = 0, \ldots, f_l(l) = 0$, $f_l(l+1) = 1$. It is obvious that $f_l(A) = \wedge_{i=0}^{l-1} a_i$, and we convert computing $[\wedge_{i=0}^{l-1} a_i]_p$ to $[f_l(A)]_p$.

We know that $f_l(x)$ can be written as $f_l(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots \alpha_l x^l \bmod p$, and thanks to the properties of shared secrets, $[f_l(x)]_p = [\alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots \alpha_l x^l \bmod p]_p$ can be simplified to $[f_l(x)]_p = \alpha_0 + \sum_{i=1}^{i=l} \alpha_i [x^i]_p$. To obtain $[f_l(A)]_p$, parties only need to compute $[A^i]_p, 1 \le i \le l$ for they have already got $\alpha_i, 0 \le i \le l$ by using Lagrange interpolation.

For $1 \le i \le l$, parties generate two shared random numbers $[b_i \in_R Z_p]_p$ and $[b_i' \in_R Z_p]_p$; see the detailed generation method in [26]. Then they compute $[B_i]_p = [b_i]_p \times [b_i']_p$ and reveal B_i:

$$\begin{aligned} \left[b_i^{-1}\right]_p [b_i]_p &= [B_i]_p \left[B_i^{-1}\right]_p, \\ \left[b_i^{-1}\right]_p [b_i]_p &= [b_i]_p [b_i']_p \left[B_i^{-1}\right]_p, \\ \left[b_i^{-1}\right]_p &= [b_i']_p B_i^{-1}. \end{aligned} \tag{23}$$

First, each party computes $[c_i']_p, 1 \le i \le l$:

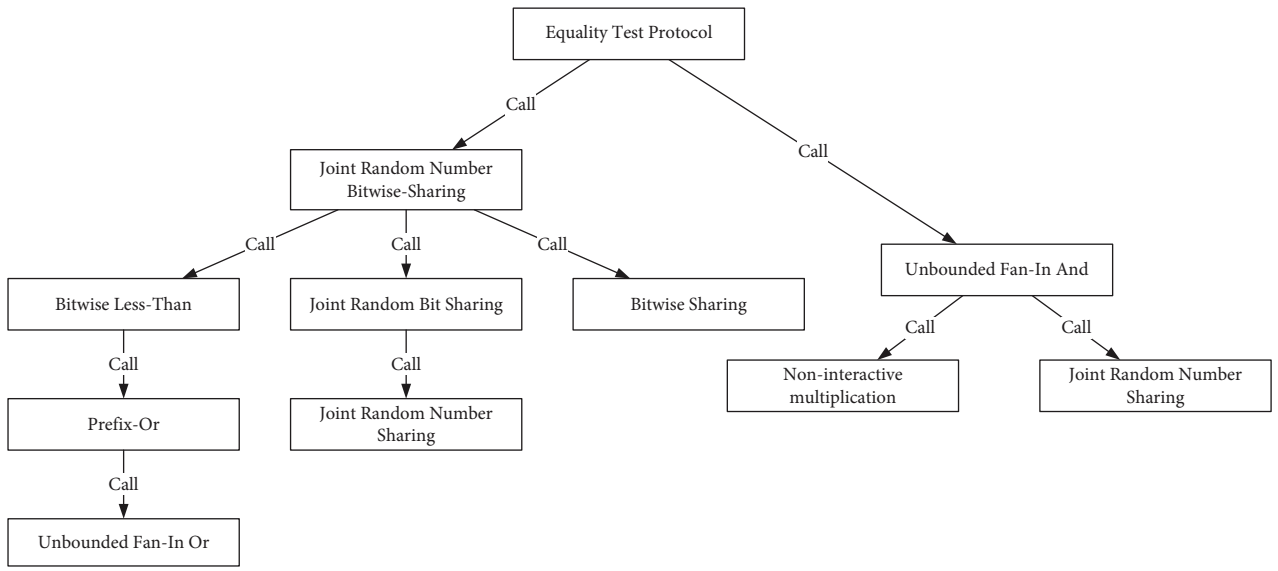

Figure 4: The call relationship of Equality Test Protocol. Subprotocols such as noninteractive multiplication are called.

Table 2: Symbolic representation of protocols.

Protocol name	Symbolic representation
Unbounded Fan-In Or	$[\vee_{i=0}^{l-1} a_i]_p$, $a_i \in \{0,1\}$
Prefix-Or	$[b_1]_p, \ldots, [b_l]_p$, $b_i = \vee_{j=1}^{i} a_j$
Bitwise Less-Than	$[a <_B b]_p$
Joint Random Number Bitwise Sharing	$[a \in_R \{0,1\}]_p$
Joint Random Number Sharing	$[r \in_R Z_p]_p$
Unbounded Fan-In And	$[\vee_{i=0}^{l-1} a_i]_p$, $a_i \in \{0,1\}$

$$
\begin{aligned}
[c_1]_p &= [A]_p \times \left[b_1^{-1}\right]_p, \\
[c_2]_p &= [A]_p \times [b_1]_p \times \left[b_2^{-1}\right]_p, \\
[c_3]_p &= [A]_p \times [b_3]_p \times \left[b_3^{-1}\right]_p, \\
&\vdots, \\
[c_l]_p &= [A]_p \times [b_{l-1}]_p \times \left[b_l^{-1}\right]_p.
\end{aligned} \tag{24}
$$

Then, they reveal c_i'. The parties can compute $[A^i]_p = (\prod_{k=1}^{i} c_k)[b_i]_p$, $1 \le i \le l$ in parallel. The reason why $A = 1 + \sum_{i=l-1}^{i=0} a_i$ is to hide the case when the number of 1 in a_i is zero.

The framework proposed by Damgård is suitable for many similar subprotocols, such as Unbounded Fan-In OR and Unbounded Fan-In XOR.

3.4. Secure UTXO Based on Secret Sharing. The challenge of the scheme is to support the privacy of the transaction; the other nodes should not know the value of the transaction except the two parties to the transaction. The transaction can be completed and verified without exposing the balance and the transaction value. Figure 5 shows a general overview of the scheme. When a transaction is initiated, the other nodes on the blockchain only need to receive the secret shares sent by both parties of the transaction, use Comparison Protocol to calculate whether the balance of A is greater than the transaction value, and use Equality Test Protocol to calculate whether the two distributed transaction values are equal. The rest of this section will describe the calculation of the entire transaction process and the changes in the secret shares held at each node before and after the transaction.

4. The Overview of Blockchain

To make the verification of balance relatively efficient in the transaction process, this scheme adopts the UTXO-based blockchain model. However, unlike the UTXO of the Bitcoin system, our UTXO holds secret shares. Since each node keeps different secret shares, so they keep different physical chains. Because different secret shares correspond to the same transaction, nodes keep the same logical chain. The corresponding relationship between physical chain and logical chain is shown in Figure 6.

4.1. Transaction Process. This section describes in detail the calculations designed during the transaction. By combining Shamir's secret sharing scheme, it realizes the confidentiality of balance and transaction value on the premise of ensuring the original properties of the blockchain.

The payer A and the payee B initiate a transaction to the blockchain. Suppose the balances of the two accounts are T_A and T_B. The transaction value which A needs to distribute is w_1, and the transaction value that B distributes is w_2. The execution of a transaction is mainly divided into two stages: (1) verify that the balance is greater than the payment, and (2) verify that the payment is equal to the value received. The transaction process is as follows.

4.1.1. Verify That the Balance T_A of A Is Greater Than the Payment w_1. The balance of A and B is stored in the form of secret shares and is expressed as $f_{T_A}(x)$ and $f_{T_B}(x)$, where

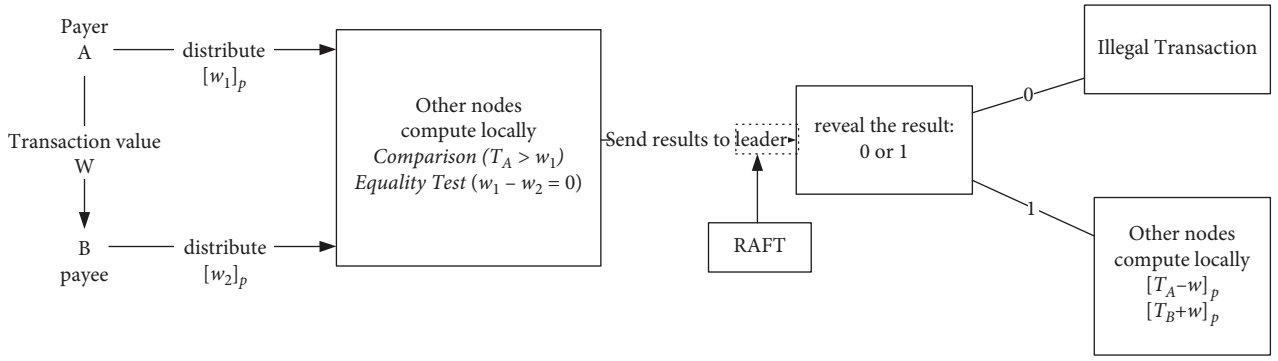

Figure 5: Transaction process. In the whole transaction process, no information about the value will be exposed, and only the comparison result 0 or 1 will be recovered.

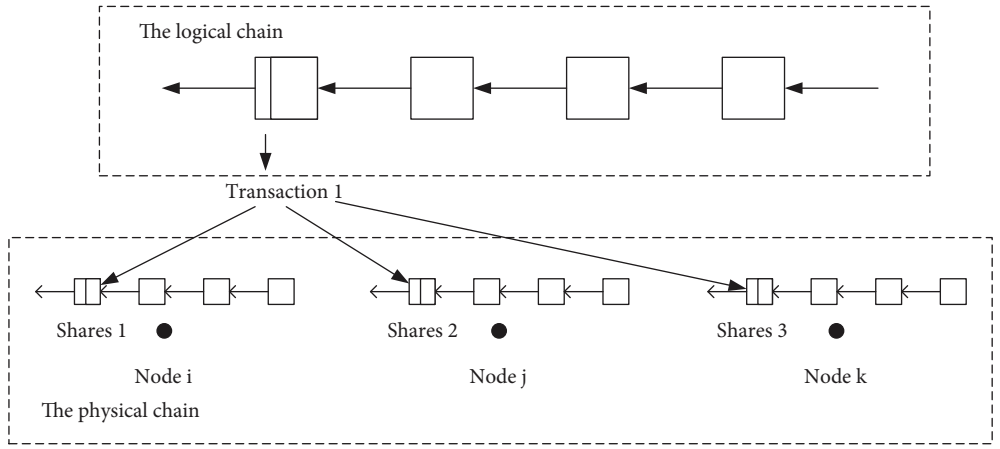

Figure 6: Logical chain and physical chain. The nodes hold different physical chains that correspond to the same logical chain.

$$f_{T_A}(x) = T_A + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \bmod p,$$
$$f_{T_B}(x) = T_B + a_1' x + a_2' x^2 + \cdots + a_{t-1}' x^{t-1} \bmod p. \quad (25)$$

When A trades, A generates a random polynomial and takes w_1 as the secret. Then A calculates the secret shares corresponding to each node and distributes them. Similarly, B also generates a random polynomial that distributes the transaction value w_2 to other nodes. The random polynomial for w_1 and w_2 is as follows:

$$f_{w_1}(x) = w_1 + b_1 x + b_2 x^2 + \cdots + b_{t-1} x^{t-1} \bmod p,$$
$$f_{w_2}(x) = w_2 + b_1' x + b_2' x^2 + \cdots + b_{t-1}' x^{t-1} \bmod p, \quad (26)$$
$$f_{w_2}(x) = w_2 + b_1' x + b_2' x^2 + \cdots + b_{t-1}' x^{t-1} \bmod p.$$

After the above operation, each node holds the secret shares of the balance of T_A and T_B and the transaction value of w_1 and w_2. We use $[w_1]_p$, $[w_2]_p$, $[T_A]_p$, and $[T_B]_p$ to represent simplicity. To compare the size of w_1 and T_A, we need to use the LSB Protocol using wraparound mentioned earlier. We can calculate $[w_1 < p/2]_p$, $[T_A < p/2]_p$, and $[T_A - [w_1 < p/2]_p \bmod p]_p$ very easily with LSB Protocol using wraparound. Then, we discuss all the cases of w_1 and T_A using $P/2$.

Case 1. If $[T_A < p/2]_p = 1$ and $[w_1 < p/2]_p = 0$, then $[T_A < w_1]_p = 1$.

Case 2. If $[T_A < p/2]_p = 0$ and $[w_1 < p/2]_p = 1$, then $[T_A < w_1]_p = 0$.

Case 3. If $[T_A < p/2]_p = 0$, $[w_1 < p/2]_p = 1$, and $[T_A - w_1 < p/2]_p = 0$, then $[T_A < w_1]_p = 1$.

Case 4. If $[T_A < p/2]_p = 0$, $[w_1 < p/2]_p = 0$, and $[T_A - w_1 < p/2]_p = 1$, then $[T_A < w_1]_p = 1$.

Case 5. If $[T_A < p/2]_p = 1$, $[w_1 < p/2]_p = 1$, and $[T_A - w_1 < p/2]_p = 0$, then $[T_A < w_1]_p = 1$.

Case 6. If $[T_A < p/2]_p = 1$, $[w_1 < p/2]_p = 1$, and $[T_A - w_1 < p/2]_p = 1$, then $[T_A < w_1]_p = 0$.

When $[w_1 < T_A]_p = 1$, it means A has enough balance to initiate the transaction.

4.1.2. Verify That the Payment w_1 is Equal to the Amount Received w_2. When a transaction is initiated, the other parties must verify that the money spent by A is equal to the money received by B. Otherwise, some malicious nodes will

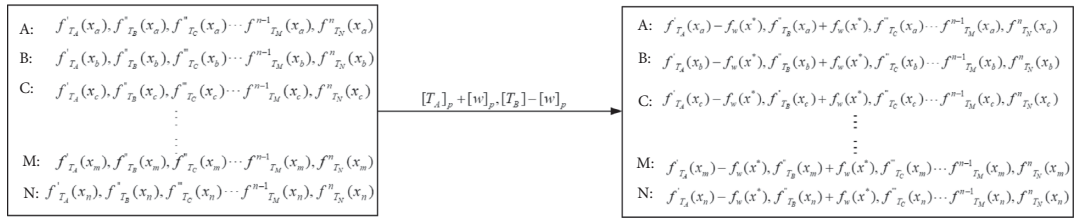

Figure 7: The change of the secret shares. In the secret shares kept by each node, only the shares related to the transaction node will change.

use the imbalance to cheat money. For example, the transaction value A that distributes to other nodes is 5 BTC, while the transaction value that distributes by B is 3 BTC. If it is not verified, 2 BTC will be defrauded by malicious nodes.

During the transaction, A and B distribute secret shares of w_1 and w_2 in parallel. We still use $[w_1]_p$ and $[w_2]_p$ for simplicity. Nodes can easily compute $[w_1 - w_2]_p$, and if they can verify $[a = w_1 - w_2 = 0]_p$, they can achieve their goals. The framework of the whole verification process is very clear. We randomize a by $c = a + r$ firstly. Then, all we have to do is verify that $c = r$ to determine $a = 0$.

First, nodes generate a bitwise shared random number by $[r \in_R Z_p]_B$ and get $[r]_p = \sum_{i=0}^{i=l-1} 2^i [r_i]_p$. Then, nodes compute $[c]_p = [a]_p + [r]_p$ and reveal $c = a + r \bmod p$. Nodes can easily obtain the binary of c for it is public. Then the parties compute $[c_i']_p$, which shows if bits of c and r are equal.

$$[c_i']_p = \begin{cases} [r_i]_p \text{if} c_i = 1, \\ 1 - [r_i]_p \text{if} c_i = 0. \end{cases} \quad (27)$$

It is obvious that $c_i' = 1$ if $c_i = r_i$. Finally, parties obtain $[a = 0]_p$ by computing the Unbounded Fan-In And $[\wedge_{i=0}^{i=l-1} c_i']_p$. Through the above steps, nodes can compute $[w_1 - w_2 = 0]_p$.

After the above two steps, the nodes hold the secret shares of $[w_1 - w_2 = 0]_p$ and $[w_1 < T_A]_p$. As long as any t nodes send the secret shares to the current leader, the result can be revealed. If the verification result is true, each node calculates $[T_A - w]_p$ and $[T_B + w]_p$ locally, and the transaction is written to the blockchain. If the verification result is false, the transaction is ruled illegal.

4.2. Changes of Secret Shares. As we described earlier, each node stores secret shares of the balance of the other nodes. The secret shares held by each node before trading are shown on the left side of Figure 7, and it is easy to see that the secret shares obtained by different nodes are different values of x calculated by the different random polynomials. After verifying the transaction, thanks to the properties of secret shares, each node can add and subtract the secret shares of the balance of A and B with the secret shares of the transaction value directly.

4.3. The Data Structure. In this section, we will describe the major data structures and the main properties, including the basic unit Transaction, the Block on which transactions are recorded, the Log held locally by the nodes, and the UTXO. We still use the block structure, but unlike the public chain, we simply keep transactions on the logical chain, and there is no amount of money involved. The local log records secret shares of each node's balance that is constantly updated. We made a slight change to the UTXO structure in Bitcoin, so instead of holding unspent Bitcoin, it is the shares of the node's most recent balance.

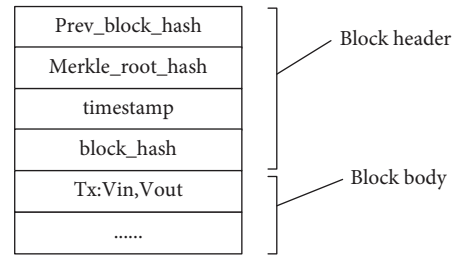

Figure 8: Block. Blocks do not contain difficulty values required for mining.

4.3.1. Block. Similar to the Bitcoin system, blocks are divided into head and body. The specific composition of the block is shown in Figure 8. However, because of the different consensus algorithms, we do not need proof of work for mining, so there are no random numbers in the block header for proof of work. It simply contains the header of the previous block, the Merkle tree hash of the current block, and the timestamp.

4.3.2. Transaction. The transaction is the basic unit that makes up the block, and it consists of two main parts: V_{in} and V_{out}. V_{in} represents the input to the transaction, which points to the most recent unspent shares of the balance in UTXO. It is important to note that UTXO here holds the shares of the balance. Even if the balance of an account is insufficient to pay, the calculated secret shares are not necessarily zero, so UTXO does not reveal the balance of an account.

As shown in Figure 9, in the case of a coinbase transaction, initial balances can be assigned to each account, and this transaction corresponds to one input and multiple outputs. To_spent in V_{in} is a pointer to UTXO, and V_{out} contains the addresses of the output and the secret shares to allocate.

4.3.3. Log. The log locally recorded by each node consists of the type, timestamp, and secret shares of the balance updated after the transaction. It can be seen from Figure 10 that each

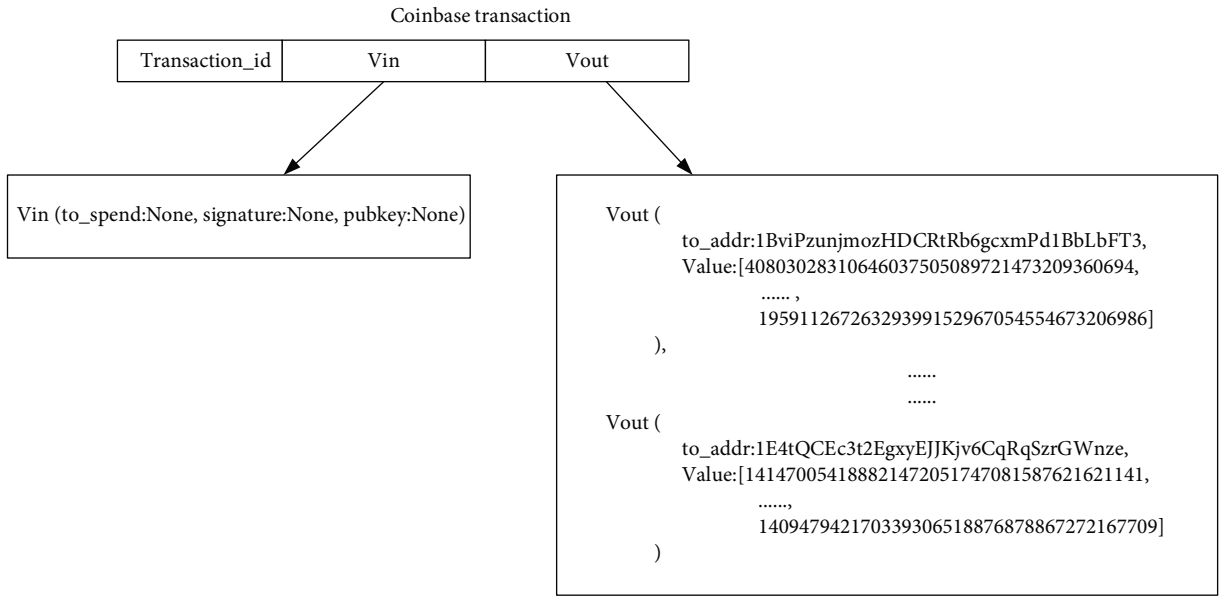

Figure 9: Transaction. V_{in} and V_{out} are in the form of secret shares rather than specific values.

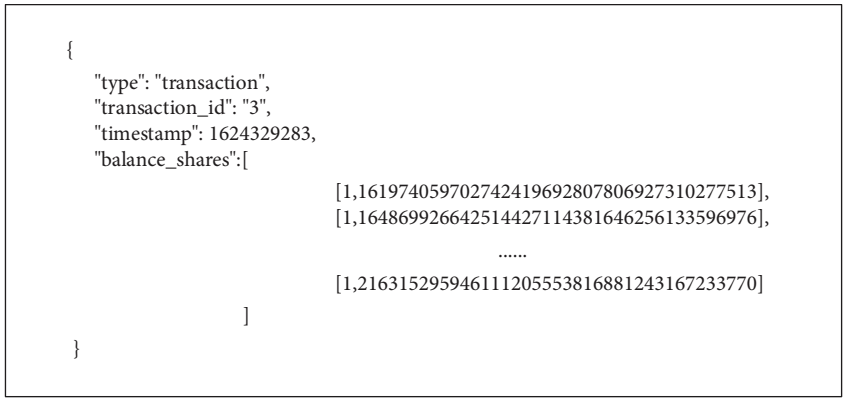

Figure 10: Log. The local storage of each node involves only secret shares.

node will hold many secret shares, which are the balances of different nodes. It should be noted that only the secret shares of balance after the transaction are kept here, and no data related to the transaction value is involved.

4.3.4. UTXO. In a nutshell, updating UTXO is simply by packing the *V* out of each transaction. As shown in Figure 11, for a verified transaction, the input points to the latest unconsumed secret shares and the output secret shares are packaged as a UTXO unit and added to UTXO_SET. We have given an example of UTXO in Figure 12 to understand the properties of UTXO. It is important to note that since there are multiple outputs from a transaction, it is necessary to specify the output of the transaction when packaging as a UTXO unit.

5. Attack Model

In the attack model of our scheme, we assume that each node is semihonest. That is, each node will correctly execute the protocol, but it may be monitored by malicious attackers to obtain its input, output, and the information obtained in the protocol operation process. The attack model in our scheme contains the following assumptions:

(1) A malicious attacker can collect the secret shares (less than t) during the protocol execution of each node and try to recover its plaintext. According to Lagrange interpolation, the protocol is unconditionally secure when the attacker cannot collect more than t secret shares.

(2) An attacker can take some nodes offline to prevent them from participating in the resulting recovery of protocol execution. Due to the threshold nature of our scheme itself, even if some nodes cannot participate in decryption, when there are t nodes online, the smooth implementation of the protocol can be realized.

(3) An attacker can collect secret shares and try to gain any information about balance and trading value.

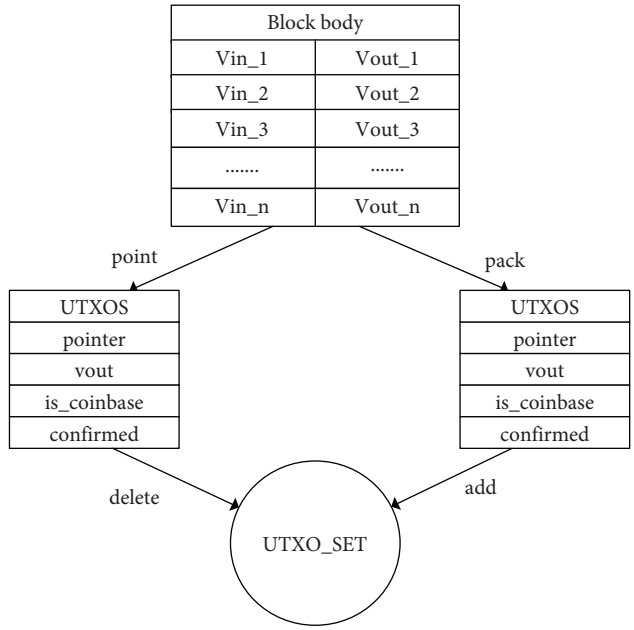

Figure 11: The process of UTXO packing. Each V_{out} is packaged into UTXO_SET to be used as V_{in} for the next transaction.

```
[
  UTXO (
        vout:Vout (
                  to_addr:1KsXfHndpbjCbZB7GGFDNwrq5KNqQYgrpd,
                  value:[1,216315295946111205553816881243167233770],
                        [2,538470078309579687795109896920228129080],
                                  .....
                  ),
        pointer:Pointer (
                        tx_id:1b67a56b6887b18ec242afc1b7e58350ea69966a7c4af34bc8a58ce9aaaa1800,
                        n:1
                      )
      )
]
```

Figure 12: UTXO. A UTXO item contains a pointer to a transaction and its V_{out}.

Since balance and transaction value are distributed in the form of shares in the whole process, even when the nodes call the Comparison Protocol, they only share the truth value of the comparison result, so as long as the attacker collects less than t shares, the attack will be invalid.

6. Analysis

6.1. Performance Analysis. The complexity of our scheme mainly contains two aspects: (1) communication complexity of making consensus on the blockchain; (2) computational complexity when validating transactions. In the second case, we can see that verifying a transaction requires a Comparison Protocol and Equality Test Protocol. The two protocols are composed of several subprotocols, and the most complicated part is calculating $[a]_p \times [b]_p$. We continue to use the same method of expressing complexity as in [13], thinking of $[a]_p \times [b]_p$ as an invocation. So the Comparison Protocol would require 15 rounds of interactions and $279l + 5$ (l is the bit length of the prime p) invocations, and the Equality Test Protocol requires 8 rounds and $81l$ invocations.

We work on a 6 Core AMD Machine with 3600 3.9 GHz CPUs and 16 GB of RAM, running on 64-bit Windows 10. The number of nodes in the blockchain is 11, and according to the two aspects of complexity, we give two figures of experimental results to describe intuitively; Figures 13 and 14 show the changes of rounds and the time of making consensus as the transaction increases, respectively. The number of interaction rounds is related to the number of transactions, and the rounds required for each transaction are reduced due to our optimization of multiplication. To simulate congestion in the real world, we set the client to sleep (2) for each round of messages sent. It can be seen from the experimental results that the time efficiency of reaching consensus is relatively high.

For the consensus protocol without privacy protection, allocating secret shares for calculation will increase the

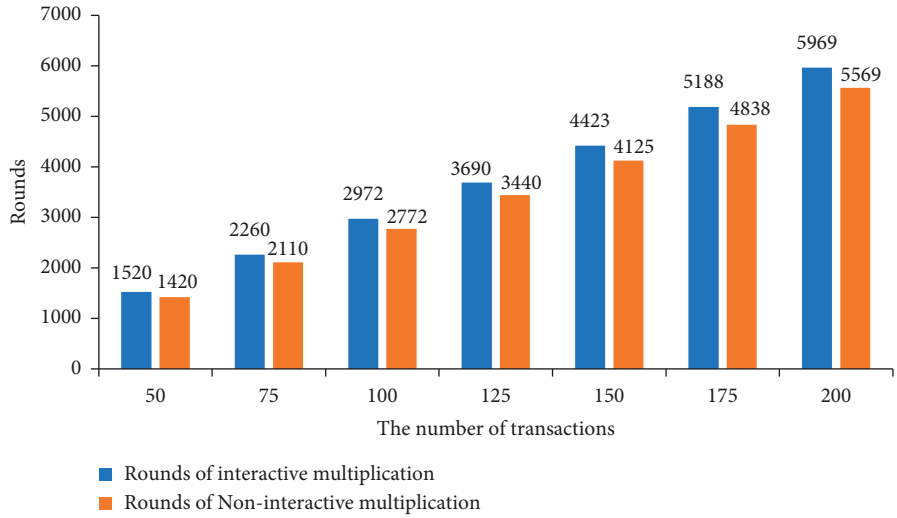

Figure 13: Changes in the number of interactive rounds. As the number of rounds in the multiplication is reduced, the change is linear.

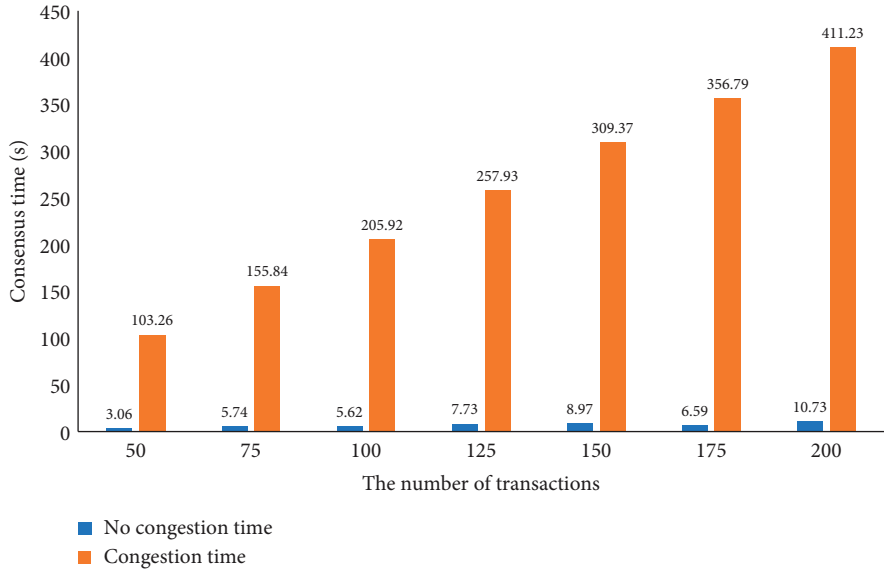

Figure 14: Change of consensus time. To prevent information loss, the interval for sending information is controlled as sleep (2).

storage of the node, but according to the privacy needs of the transaction, the increased storage is worthwhile. The comparison of the storage of our scheme with that of the nonprivacy scheme is shown in Figure 15.

Since we proposed the noninteractive multiplication, the communication cost of a single node also decreased significantly. For the interactive multiplication, the nodes need to multiply the shares received and distribute them again, resulting in the fact that the communication volume of a single node is $2\log_2 p + n\log_2 p$; however, noninteractive multiplication reduces this to $2\log_2 p$.

6.2. Security Analysis. To formally analyse the security of multiparty computing protocols, Golderich et al. [27] introduced a realistic/ideal model security analysis method. The multiparty computational cryptographic protocol is secure if the view obtained by the PPT adversary attacking the actual protocol is indistinguishable from the view obtained by attacking the ideal model. In the ideal model, there is a trusted third party trusted by all participants. Participants do not interact directly with each other but interact with the trusted third party. The trusted third party calculates the ideal function using the input of participants and returns the results to participants. In the ideal model, the protocol is a security protocol, and the effect of the adversary attacking the ideal model is negligible. If an adversary cannot distinguish whether its attack is against the ideal model or the real model, it means that the effect of attacking the actual protocol is also negligible, so the protocol is secure.

When security analysis is considered, it is usually for a single protocol in a specific environment, but the actual cryptographic protocols are often executed concurrently or in combination with each other. The external environment

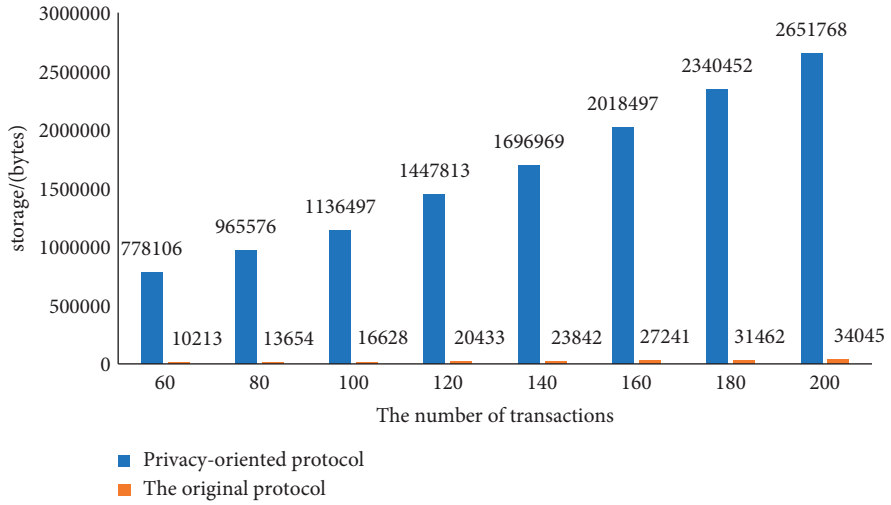

Figure 15: Storage comparison. As the value is divided into multiple secret shares, the storage cost of nodes increases.

of protocol execution can be summarized as an independent model, concurrent model, general combinable model, and so on. Canetti proposed the UC framework [28]. The biggest feature of security in the UC framework is that a secure protocol is still secure when it is combined with any other protocol or as a component of a larger protocol.

UC framework also applies the ideal/reality model, and the difference is that the UC framework analyses protocol security from the perspective of the environment.

The framework abstracts the ideal protocol π into a secure ideal function $\mathcal{F}$. The ideal world and the real world are in the same environment $\mathcal{Z}$. Participants and attackers in the ideal world interact with the ideal function $\mathcal{F}$, and they cannot communicate directly with each other. The ideal function $\mathcal{F}$ is essentially an unassailable trusted third party.

Definition 1. When any real adversary $\mathcal{A}$ has a simulated adversary $\mathcal{S}$ in the ideal world, the probability that environment $\mathcal{Z}$ can distinguish between the real adversary $\mathcal{A}$ interacting with protocol π and the simulated adversary $\mathcal{S}$ interacting with the ideal function $\mathcal{F}$ is negligible, and the protocol π is said to securely implement the ideal function $\mathcal{F}$.

Theorem 1. *If the protocol ρ securely implements the ideal function $\mathcal{F}$, π is the protocol under hybrid model $\mathcal{F}-$. Then, the combined protocol $\pi^{\rho/f}$ replacing the ideal function $\mathcal{F}$ in protocol π with protocol ρ is also UC security.*

Our protocol π is mainly composed of $\pi_{equality}$, $\pi_{comparison}$, and $\pi_{addition}$. To prove that the protocol π is secure for UC, it is necessary to prove that $\pi_{equality}$, $\pi_{comparison}$, and $\pi_{addition}$ are secure for UC. That is, the three protocols securely implement the ideal functions $\mathcal{F}_{equality}$, $\mathcal{F}_{comprison}$, and $\mathcal{F}_{addition}$, respectively, and then prove that π is secure for UC under $\mathcal{F}_{equality}\mathcal{F}_{comprison}\mathcal{F}_{addition}$-hybrid model.

6.2.1. Security Analysis of $\pi_{equality}$. In π_{equality}, each participant p_i holds secret shares $f_a(ID_i)$ and $f_b(ID_i)$ of secrets a and b. Participants calculate the secret shares of Boolean results $f_{(a=b)}(ID_i)$ by the protocol to estimate whether a and b are equal and do not expose the comparison results of a and b. In the protocol, Unbounded Fan-In And is called π_{and}. We first proved the security of π_{and}.

Claim 1. The Unbounded Fan-In And protocol is hybrid-model security when the adversary corrupts no more than t participants.

We describe the ideal function of π_{and} as follows.

6.2.2. Ideal Function $\mathcal{F}_{and}$

$n = 0$, where n is used to record the number of participants corrupted by the adversary.

(1) When it receives the input $(\text{input}, \text{sid}, p_i, ([a_{l-1}]_p, \ldots, [a_0]_p))$ from participant p_i, it computes $[A]_p = [a_{l-1}]_p + \cdots + [a_0]_p + 1$, uses Lagrange interpolation to determine a polynomial $f_l(x)$ of order l which is satisfying $f_l(2k) = 0$, $f_l(2k+1) = 1$, $k = 0, 1, 2 \ldots$, and sends $[A]_p$ to participants.

(2) When it receives $(\text{compute}, \text{sid}, p_i, ([c_l]_p, \ldots, [c_1]_p), [b_j]_p)$ from participant p_i, it first recovers $c_1, \ldots, c_l$ using Lagrange interpolation and then computes $[A^j]_p = \prod_{k=1}^{j} c_k [b_j]_p$. It substitutes $[A^j]_p$, $j \in (1, l)$ into $f_l(x)$ to calculate $[f_l(A)]_p$. The two inputs of p_i are written as x_i, and the output in (1) is y_{1i}, $y_{2i} = [f_l(A)]_p$.

(3) When it receives a request $(\text{output}, \text{sid}, p_i)$ from participant p_i. First, it determines whether p_i is corrupted. If not, it returns y_{2i}; if so, it sets $n+ = 1$. If $n > t$, then it produces a delayed output y_{2i} to p_i.

(4) When it receives $(\text{corrupted\_input}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted, and x_i and y_{1i} are sent to $\mathcal{S}$.

(5) When it receives $(\text{corrupted\_output}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted, and $f_l(A)$ is recovered using Lagrange

interpolation using y_{2i} of all participants marked with corrupted. If a valid solution can be successfully recovered, let $y_{2i} = f_l(A)$ of all corrupted participants and send y_{2i} and $f_l(A)$ of the current participant to $\mathcal{S}$; otherwise, only y_{2i} of the current participant is sent.

In an ideal world, when the participant p_i receives $[A]_p$ in the interaction with the ideal function, it will interact with the environment machine $\mathcal{Z}$, which needs to simulate inverse and multiplication protocols to calculate $[c_l]_p, \ldots, [c_1]_p$ and generate $[b_j]_p$ for the participant. From the definition of the UC framework, we know that this process is ideal and secure. Next, we prove the security of π_{And} from the view of environment machine $\mathcal{Z}$.

Proof. Given an adversary in the real world, construct an adversary $\mathcal{S}$ in the ideal world such that the probability of any environmental machine $\mathcal{Z}$ distinguishing between the two worlds is negligible. The ideal adversary $\mathcal{S}$ forwards any input from the environment to $\mathcal{A}$, and any output of $\mathcal{A}$ is regarded as the output of $\mathcal{S}$. The specific operation of $\mathcal{S}$ is as follows:

(1) Simulation establishment.

When x_i, y_{1i}, y_{2i} belonging to p_i are received from the ideal function $\mathcal{F}_{\text{and}}$, the simulated adversary gets x_i, y_{1i}, y_{2i}.

(2) Simulate corruption.

When a real-world adversary $\mathcal{A}$ corrupts p_i, $\mathcal{S}$ corrupts p_i by calling Corrupted_input or Corrupted_output and forwards x_i, y_{1i}, y_{2i} to $\mathcal{A}$.

(3) Indistinguishability.

Define Event. When $n \le t$, the value of y_{2i} received from the ideal function $\mathcal{F}_{\text{and}}$ is a bit 0 or 1, which is the recovered value of *and*.

According to Lemma 1, the event is impossible to occur. So the two worlds are indistinguishable. Claim 1 is proved. □

Theorem 2. *When the adversary $\mathcal{A}$ corrupts no more than t participants, the protocol $\pi_{equality}$ securely implements the ideal function $\mathcal{F}_{equality}$ under the $\mathcal{F}_{and}$-hybrid model.*

The ideal function of π_{equality} is as follows:

Ideal function $\mathcal{F}_{\text{equality}}$.

$n = 0$, where n is used to record the number of participants corrupted by the adversary.

(1) When it receives the input $(\text{input}, \text{sid}, p_i, [a]_p, [b]_p, [r \in_R \mathbb{Z}_p]_B, [r]_p)$ from participant p_i, then it computes $[x]_p = [a]_p - [b]_p$ and $[c]_p = [x]_p + [r]_p$. After receiving input from at least t participants, using the Lagrange interpolation to recover c, and calculating the bit form of $(c_{l-1}, \ldots, c_0)$ and $[c'_j]_p = \begin{cases} [r_j]_p \text{if} r_j = 1 \\ 1 - [r_j]_p \text{if} r_j = 0 \end{cases}$, $j = 0, 1, 2\ldots$, then it sends $[c'_j]_p$ to participants.

(2) When it receives $(\text{input}, \text{sid}, p_i, [f_l(A)]_p)$ from participant p_i, the two inputs of p_i are written as x_i, and the output in (1) is y_{1i}, $y_{2i} = [f_l(A)]_p$.

(3) When it receives a request $(\text{output}, \text{sid}, p_i)$ from participant p_i, first it determines whether p_i is corrupted. If not, it returns y_{2i}; if so, it sets $n+ = 1$. If $n > t$, then it produces a delayed output y_{2i} to p_i.

(4) When it receives $(\text{corrupted\_input}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted, and x_i, y_{1i} are sent to $\mathcal{S}$.

(5) When it receives $(\text{corrupted\_output}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted, and $f_l(A)$ is recovered using y_{2i} of all participants marked with corrupted by Lagrange interpolation. If a valid solution can be successfully recovered, let $y_{2i} = f_l(A)$ of all corrupted participants and send the y_{2i} and $f_l(A)$ of the current participant to $\mathcal{S}$; otherwise, only y_{2i} of the current participant is sent.

In an ideal world, the participant p_i first interacts with the environment machine $\mathcal{Z}$ to obtain $[r \in_R \mathbb{Z}_p]_B$, $[r]_p$, and when receiving $[c'_j]_p$ from the ideal function, $\mathcal{F}_{\text{equality}}$, it interacts with the ideal function $\mathcal{F}_{\text{and}}$ to compute the secret shares of and.

Proof. Given an adversary in the real world, construct an adversary $\mathcal{S}$ in the ideal world such that the probability of any environmental machine $\mathcal{Z}$ distinguishing between the two worlds is negligible. The ideal adversary $\mathcal{S}$ forwards any input from the environment to $\mathcal{A}$, and any output of $\mathcal{A}$ is regarded as the output of $\mathcal{S}$. The specific operation of $\mathcal{S}$ is as follows:

(1) Simulation establishment.

When x_i, y_{1i}, y_{2i} belonging to p_i are received from the ideal function $\mathcal{F}_{\text{equality}}$, the simulated adversary gets x_i, y_{1i}, y_{2i}.

(2) Simulate corruption.

When a real-world adversary $\mathcal{A}$ corrupts p_i, $\mathcal{S}$ corrupts p_i by calling Corrupted_input or Corrupted_output and forwards x_i, y_{1i}, y_{2i} to $\mathcal{A}$.

(3) Indistinguishability.

Define event 1: when $n \le t$, the value of y_{2i} received from the ideal function $\mathcal{F}_{\text{equality}}$ is a bit 0 or 1, which is the recovered value of *and*. According to Lemma 1, the event is impossible to occur. So the two worlds are indistinguishable. Claim 1 is proved.

Define event 2: for $(\text{input}, \text{sid}, p_i, [f_l(A)]_p)$ where $[f_l(A)]_p$ is exactly equal to $f_l(A)$, when the adversary corrupts no more than t participants, according to Claim 1, the event cannot occur. So the two worlds are indistinguishable. Theorem 2 is proved.

(2) Security Analysis of $\pi_{\text{comparison}}$. In $\pi_{\text{comparison}}$, each participant p_i holds secret shares $f_a(ID_i)$ and $f_b(ID_i)$ of a and b. The participants calculate the secret shares of the

comparison result $f_{(a<b)}(ID_i)$ and do not expose a and b themselves. $[a < p/2]_p$, $[b < p/2]_p$, and $[a - b \bmod p < p/2]_p$ are computed by calling the LSB Protocol, so we need to prove the security of π_{LSB}. □

Claim 2. The LSB Protocol is hybrid-model security when no more than t participants are corrupted.

To prove the above claim, we first describe the behavior of adversary, environment machines in the ideal world of the UC framework, and the ideal function of the LSB Protocol. Since the environment machine $\mathcal{Z}$ is used to simulate the external environment of protocol operation, we assume that the environment machine $\mathcal{Z}$ simulates the Joint Random Number Bitwise Sharing, Bitwise Less-Than, and noninteractive multiplication protocol that need to be called in the LSB Protocol.

The participants can communicate with the environment machine $\mathcal{Z}$, and its specific behavior is as follows:

Environment machine $\mathcal{Z}$:

(1) Initialization phase: simulate Joint Random Number Bitwise Sharing to generate $[r \in_R \mathbb{Z}_p]_B$ and $[r]_p$ for each participant p_i.

(2) Interaction phase: when the input $(\text{compute}, \text{sid}, c, [c]_p, (c)_0 \oplus [(r)_0]_p)$ is received from the participant p_i, simulates the Bitwise Less-Than to produce an output $[c <_B r]_p$, and simulates the noninteractive multiplication protocol to produce an output $2[c <_B r]_p \times \{(c)_0 \oplus [(r)_0]_p\}$, then send $([c <_B r]_p, 2[c <_B r]_p \times \{(c)_0 \oplus [(r)_0]_p\})$ to p_i.

The behavior of the participant p_i is described as follows:

Participant p_i:

(1) When the outputs $[r \in_R \mathbb{Z}_p]_B$ and $[r]_p$ are received from the environment machine $\mathcal{Z}$, $(\text{input}, sid, p_i, [x]_p, [r \in_R \mathbb{Z}_p]_B, [r]_p)$ is sent to the ideal function $\mathcal{F}_{\text{LSB}}$.

(2) When the output $(c, [c]_p, (c)_0 \oplus [(r)_0]_p)$ is received from the ideal function $\mathcal{F}_{\text{LSB}}$, $(\text{compute}, \text{sid}, c, [c]_p, (c)_0 \oplus [(r)_0]_p)$ is sent to the environment machine $\mathcal{Z}$.

(3) When the output $([c <_B r]_p, 2[c <_B r]_p \times \{(c)_0 \oplus [(r)_0]_p\})$ is received from the environment machine $\mathcal{Z}$, $(\text{compute}, \text{sid}, p_i, [c <_B r]_p, 2[c <_B r]_p \times \{(c)_0 \oplus [(r)_0]_p\})$ is sent to the ideal function $\mathcal{F}_{\text{LSB}}$.

(4) Send the output $(\text{output}, \text{sid}, p_i)$ to the ideal function $\mathcal{F}_{\text{LSB}}$.

The ideal function $\mathcal{F}_{\text{LSB}}$ is described as follows:

Ideal function $\mathcal{F}_{\text{LSB}}$:

$n = 0$, where n is used to record the number of participants corrupted by the adversary.

(1) When it receives the input $(\text{input}, \text{sid}, p_i, [x]_p, [r \in_R \mathbb{Z}_p]_B, [r]_p)$ from the participant p_i, it computes $[c]_p = [x]_p + [r]_p$, and when it receives inputs from at least t participants, it recovers c using Lagrange interpolation and sends $(c, [c]_p, (c)_0 \oplus [(r)_0]_p)$ to the participants.

(2) When it receives $(\text{compute}, \text{sid}, p_i, [c <_B r]_p, 2[c <_B r]_p \times \{(c)_0 \oplus [(r)_0]_p\})$ from participant p_i, compute $([(x)_0]_p = [c <_B r]_p + \{(c)_0 \oplus [(r)_0]_p\} - 2[c <_B r]_p \times \{(c)_0 \oplus [(r)_0]_p\})$. The two inputs of p_i are written as x_i, and the output in (1) is y_{1i}, $y_{2i} = [(x)_0]_p$.

(3) When it receives a request $(\text{output}, \text{sid}, p_i)$ from participant p_i, first, it determines whether p_i is corrupted. If not, it returns y_{2i}; if so, it sets $n+ = 1$. If $n > t$, then it produces a delayed output y_{2i} to p_i.

(4) When it receives $(\text{corrupted\_input}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted, and x_i and y_{1i} are sent to $\mathcal{S}$.

(5) When it receives $(\text{corrupted\_input}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted, and $(x)_0$ is recovered using y_{2i} of all participants marked with corrupted by Lagrange interpolation. If a valid solution can be successfully recovered, let $y_{2i} = (x)_0$ of all corrupted participants and send y_{2i} and $(x)_0$ of the current participant to $\mathcal{S}$; otherwise, only y_{2i} of the current participant is sent.

The proof for Claim 2 is similar to Claim 1.

The computation steps except π_{LSB} in $\pi_{\text{comparison}}$ are noninteractive; participants can complete them locally without any communications, so there is no adversary attack, and it is as secure as π_{LSB}.

(3) Security Analysis of π_{additoin}. Since π_{addition} itself is noninteractive, participants can complete without any communication in the local, and there is no adversary attack, so the protocol π_{addition} is unconditional security.

(4) Security Analysis of Our Scheme.

Ideal function $\mathcal{F}$:

$n = 0$, where n is used to record the number of participants corrupted by the adversary.

(1) When it receives the inputs $(\text{input}, \text{sid}, A, w_A, T_a)$ and $(\text{input}, \text{sid}, B, w_b, T_b)$ from transaction nodes A and B, it generates secret shares $[w_A]_p$, $[w_B]_p$, $[T_A]_p$, $[T_B]_p$ for all nodes p_i participating in the verification and sends corresponding secret shares to participants.

(2) When it receives $(\text{input}, \text{sid}, p_i, [w_A = w_B]_p, [T_A < w_B]_p)$ from participants p_i with more than t, it uses Lagrange interpolation to recover the values of $w_A = w_B$ and $T_A < w_B$. If $(T_A < w_B) = 0$ and $(w_A = w_B) = 1$, the values of $w_A = w_B$ and $T_A < w_B$ are sent to the leader, and the corresponding secret shares of each participant are used to calculate $[T_A - w_B]_p$ and $[T_B + w_A]_p$, respectively. Otherwise, the values of $w_A = w_B$ and $T_A < w_B$ are sent to the leader, and the protocol process is terminated. Let $y_i = ([T_A - w_B]_p, [T_B + w_A]_p)$, and x_i is the set of all inputs for p_i.

(3) When it receives a request $(\text{output}, \text{sid}, p_i)$ from participant p_i. First, it determines whether p_i is corrupted. If not, it returns y_i; if so, it sets $n+ = 1$. If $n > t$, then it produces a delayed output y_i to p_i. When it receives $(\text{corrupted\_input}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted, and x_i is sent to $\mathcal{S}$.

(4) When it receives $(\text{corrupted\_output}, \text{sid}, p_i)$ from adversary $\mathcal{S}$, the corresponding record of p_i is corrupted. $(T_A - w_B)$, $(T_B + w_A)$ are recovered using y_i of all participants marked with corrupted by Lagrange interpolation. If a valid solution can be successfully recovered, let $y_i = (T_A - w_B)$, $(T_B + w_A))$ of all corrupted participants and send $(T_A - w_B)$, $(T_B + w_A)$ to $\mathcal{S}$; otherwise, only y_i of the current participant is sent.

Theorem 3. *Our protocol π securely implements ideal function $\mathcal{F}$ under $\mathcal{F}_{equality}, \mathcal{F}_{comprison}, \mathcal{F}_{addition}$-hybrid model.*

In an ideal world, the participant p_i first interacts with $\mathcal{F}_{\text{equality}}$ using $[w_A]_p$, $[w_B]_p$, gets its output as the value of $[w_A = w_B]_p$, and interacts with $\mathcal{F}_{\text{comparison}}$ using w_B, T_a to get its output as the value of $[T_A < w_B]_p$.

Proof. Given an adversary in the real world, construct an adversary $\mathcal{S}$ in the ideal world such that the probability of any environmental machine $\mathcal{Z}$ distinguishing between the two worlds is negligible. The ideal adversary $\mathcal{S}$ forwards any input from the environment to $\mathcal{A}$, and any output of $\mathcal{A}$ is regarded as the output of $\mathcal{S}$. The specific operation of $\mathcal{S}$ is as follows:

(1) Simulation establishment.

When x_i, y_i belonging to p_i are received from ideal function $\mathcal{F}$, the simulated adversary gets x_i, y_i.

(2) Simulate corruption.

When a real-world adversary $\mathcal{A}$ corrupts p_i, $\mathcal{S}$ corrupts p_i by calling Corrupted_input or Corrupted_output and forwards x_i, y_i to $\mathcal{A}$.

(3) Indistinguishability.

Define event 1: when $n \le t$, the value $(T_A - w_B)$, $(T_B + w_A)$ can be received from the ideal function $\mathcal{F}$. According to Lemma 1, the event is impossible to occur.

Define event 2: for $(\text{input}, \text{sid}, p_i, [T_A < w_B]_p)$ where the value of $[w_A = w_B]_p$ or $[T_A < w_B]_p$ is exactly a definite bit 0 or 1, when the adversary corrupts no more than t participants, according to Claim 1 and Claim 2, the event cannot occur. So the two worlds are indistinguishable. Theorem 3 is proved. □

7. Conclusions

This paper proposes a private-oriented transaction scheme based on the blockchain via Shamir's secret sharing. Taking advantage of the secret sharing feature, we creatively design a physical-logical chain; the characteristic is that each node maintains a physical chain locally, and physical chains map to the same logical chain. In this paper, we use the Equality Test Protocol, Comparison Protocol, and raft consensus algorithm to achieve legal transactions without disclosing any information about balance and trading value. And we propose noninteractive multiplication, which improves the efficiency of the whole scheme and reduces the risks brought by the interactive process.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest in this work.

Acknowledgments

This work was supported by the "Chengdu-Chongqing Economic Circle Construction" Scientific and Technological Innovation Project of Chongqing Municipal Education Commission under Grant KJCX2020033, the National Natural Science Foundation of China under Grant 61903053, and the Opening Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security under Grant AGK2020006.

References

[1] K. Peng, "Security challenges and opportunities for smart contracts in Internet of Things: a survey," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12004–12020, 2021.

[2] C. Feng, "Blockchain-empowered decentralized horizontal federated learning for 5G-enabled UAVs," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3582–3592, 2021.

[3] J. Wang, L. Wu, K. K. R. Choo, and D. He, "Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1984–1992, 2020.

[4] B. Bhushan and N. Sharma, "Transaction privacy preservations for blockchain technology," *International Conference on Innovative Computing and Communications*, pp. 377–393, Springer, Singapore, 2021.

[5] D. Hopwood, *Zcash Protocol Specification*, GitHub, San Francisco, CA, USA, 2016.

[6] J. Benet and N. Greco, *Filecoin: A Decentralized Storage Network" Protoc. Labs*, pp. 1–36, 2018.

[7] J. Lu, "Towards fairness-aware time-sensitive asynchronous federated learning for critical Energy infrastructure," *IEEE Transactions on Industrial Informatics*, vol. 18, 2021.

[8] P. Radoglou-Grammatikis, "Modeling, detecting, and mitigating threats against industrial healthcare systems: a combined software defined networking and reinforcement learning approach," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 2041–2052, 2021.

[9] Z. Wang, X. Pang, Y. Chen et al., "Privacy-preserving crowdsourced statistical data publishing with an untrusted server," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1356–1367, 2019.

[10] C. Wang, "Safeguarding cross-silo federated learning with local differential privacy," *Digital Communications and Networks*, Vol. 8, no. 4, pp. 446–454, 2021.

[11] B. Liu, D. Jia, J. Wang, K. Lu, and L. Wu, "Cloud-assisted safety message dissemination in VANET–cellular heterogeneous wireless network," *IEEE Systems Journal*, vol. 11, no. 1, pp. 128–139, 2017.

[12] L. Wu, Y. Zhang, K. K. R. Choo, and D. He, "Efficient and secure identity-based encryption scheme with equality test in cloud computing," *Future Generation Computer Systems*, vol. 73, pp. 22–31, 2017.

[13] Su Liu, J. Yu, X. Deng, and S. Wan, "FedCPF: an efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1616–1629, 2022.

[14] G. Zyskind and O. Nathan, "Decentralizing privacy: using blockchain to protect personal data," in *Proceedings of the 2015 IEEE Security and Privacy Workshops*, pp. 180–184, IEEE, CA, USA, May 2015.

[15] F. Benhamouda, C. Gentry, and S. Gorbunov, "Can a public blockchain keep a secret?" *Theory of Cryptography Conference*, pp. 260–290, Springer, Cham, 2020.

[16] R. K. Raman and L. R. Varshney, "Dynamic distributed storage for scaling blockchains," 2017, https://arxiv.org/abs/1711.07617arXiv preprint arXiv:1711.07617.

[17] S. Bartolucci, P. Bernat, and D. Joseph, "SHARVOT: secret SHARe-based VOTing on the blockchain," in *Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain*, pp. 30–34, ACM, Gothenburg, Sweden, May 2018.

[18] B. K. Zheng, L. H. Zhu, M. Shen et al., "Scalable and privacy-preserving data sharing based on blockchain," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 557–567, 2018.

[19] Y. Kim, R. K. Raman, Y. S. Kim, L. R. Varshney, and N. R. Shanbhag, "Efficient local secret sharing for distributed blockchain systems," *IEEE Communications Letters*, vol. 23, no. 2, pp. 282–285, 2019.

[20] M. Fukumitsu, S. Hasegawa, and J. Iwazaki, "A proposal of a secure P2P-type storage scheme by using the secret sharing and the blockchain," in *Proceedings of the IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pp. 803–810, IEEE, Gothenburg, Sweden, May 2017.

[21] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," *International Workshop on Public Key Cryptography*, pp. 343–360, Springer, Berlin, Heidelberg, 2007.

[22] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[23] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography," in *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, pp. 101–111, ACM, Puerto Vallarta Mexico, June 1998.

[24] S. Wan, M. Li, G. Liu, and C. Wang, "Recent advances in consensus protocols for blockchain: a survey," *Wireless Networks*, vol. 26, no. 8, pp. 5579–5593, 2020.

[25] I. Damgård, M. Fitzi, and E. Kiltz, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," *Theory of Cryptography Conference*, pp. 285–304, Springer, Berlin, Heidelberg, 2006.

[26] J. Bar-Ilan and D. Beaver, "Non-cryptographic fault-tolerant computing in constant number of rounds of interaction," in *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pp. 201–209, ACM, Edmonton Alberta Canada, June 1989.

[27] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pp. 307–328, 2019.

[28] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," in *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pp. 136–145, IEEE, CA, USA, October 2001.