IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 14, 2020, accepted May 7, 2020, date of publication May 11, 2020, date of current version May 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993759

MBFT: A New Consensus Algorithm for Consortium Blockchain

MINGXIAO DU, QIJUN CHEN, (Senior Member, IEEE), AND XIAOFENG MA
Department of Control Science and Engineering, Tongji University, Shanghai 200092, China

Corresponding author: Xiaofeng Ma (xiaofengma@tongji.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61733013 and Grant 61673300, and in part by the Science and Technology Commission of Shanghai Municipality under Grant 18DZ1200804

ABSTRACT Blockchain technology is an emerging distributed ledger technology that has exploratory applications in many areas. The consensus algorithm, as the core module of the blockchain, has an important impact on the security, scalability, and efficiency of the blockchain network. The consensus algorithm is also a popular topic in current blockchain technology research. In existing consortium blockchains and public blockchains, the blockchain has low efficiency or poor fault tolerance because of the limitations of the consensus algorithm. To both ensure the fault tolerance of the blockchain and improve the scalability and throughput, in this paper, we propose a new type of consensus algorithm: mixed Byzantine fault tolerance (MBFT). MBFT uses sharding and layered technology. MBFT functionally partitions the nodes that participate in the consensus process and improves the scalability and efficiency without sacrificing security. MBFT also introduces a random node selection mechanism and a credit mechanism to improve security and fault tolerance. We analyze the security and experiment on transaction throughput in a real network environment. The results prove that MBFT has good security and scalability and high throughput.

INDEX TERMS Blockchain, consensus algorithm, Byzantine fault tolerance, distributed system.

I. INTRODUCTION

The consensus algorithm is the core technology of the blockchain. The consensus algorithm is a set of mechanisms that are designed to ensure the accuracy and consistency of the data stored. To achieve lower transaction verification delays and higher fault tolerance, the consensus algorithm used on the blockchain is primarily determined by the requirements of the service and performance. In traditional distributed systems, the most commonly used consensus algorithm is based on Paxos [1]. This type of algorithm can quickly complete data synchronization in distributed systems with a limited number of nodes and tolerate crash faults. Specifically, the traditional distributed system does not need to consider that malicious nodes could tamper with the data and only needs to tolerate the fault of a portion of nodes having downtime. However, in a blockchain, the situation is more complicated. The number of nodes participating in accounting in the blockchain system is large, and the nodes do not trust each other. Some of the accounting nodes may be controlled and may tamper with the data or deliberately destroy the entire system by sending an error message [2]. In this case, the consensus algorithm employed by the blockchain system needs to have Byzantine fault tolerance [3]. The earliest consensus mechanism used in the blockchain was Proof of Work (PoW), which is adopted by Bitcoin [4]. The core idea of PoW is to allocate billing rights and billing rewards according to computing power. In the PoW blockchain system, when the computing power is sufficiently large, the system can obtain sufficient security and availability.

As an emerging technology, blockchain technology contains many existing computer technologies such as point-to-point communication, consensus algorithms, distributed storage, and encryption algorithms. Given its rapid development, blockchain technology has become increasingly widely used. The blockchain is considered a disruptive, innovative business model, similar to mainframes, personal computers, the Internet and cloud computing, which triggered technological innovation [5]. The application of the blockchain has extended to the finance industry, the Internet of Things, medical care, supply chain management, digital asset trading, property right protection and many other fields [6]–[9].

The associate editor coordinating the review of this manuscript and approving it for publication was Shiqiang Wang.

However, blockchain technology also suffers from many problems such as inadequate transactions per second (TPS), fast anonymous transactions, and decentralization. The current blockchain's TPS is relatively low because the consensus process takes a long time. For example, in a blockchain that uses the PoW consensus algorithm, all nodes need to perform hash calculations and compete for package blocks. To improve the TPS, new technologies, such as sharding technology, are being adopted in the blockchain. Sharding technology was originally used for databases. The data in a database are cut into multiple parts and then stored in multiple servers; this can improve the search performance of the server. In the blockchain, sharding technology is a mechanism for assigning transactions to different consensus groups and ultimately summarizing all the results. Elastico, which was proposed by Luu *et al.*, first used sharding technology [10]. Subsequently, Kogias *et al.* proposed Byzcoin [11], therein benefiting from an adaptation of practical Byzantine fault tolerance (PBFT) [12] and decoupling the election of a new leader from transaction verification, an approach inspired by Bitcoin-NG [13]. The Zilliqa proposed in 2017 inherits the characteristics of Elastico and combines the PoW and PBFT algorithms. Kokoris-Kogias *et al.* optimized the basis of Byzcoin and merged protocols, such as RandHound, therein proposing OmniLedger [14]. These algorithms retain the PoW consensus, require a certain amount of hashing power to ensure security, and need to solve the problems of distrust and competition among nodes. The transaction confirmation delay is large. Moreover, due to the openness of the public blockchain, the system must adopt an economic incentive to attract nodes to participate in transaction verification. Consequently, these algorithms are not very suitable for consortium blockchain.

The consensus mechanism based on PBFT is currently mostly used in the consortium blockchain. This makes it necessary to communicate multiple times between two nodes to confirm a transaction. These communications can well maintain the consistency of the results; however, when the number of nodes increases, the speed of synchronization is severely decreased. Therefore, this paper proposes a new mixed Byzantine fault tolerance consensus algorithm (MBFT) based on the consensus algorithm that we have introduced briefly in 2018 [15]. MBFT improves the scalability and performance while ensuring fault tolerance. The main innovations of this paper are as follows:

1) We use sharding technology and process a two-layer consensus algorithm. In MBFT, we assign nodes to different consensus groups and design the new transaction verification and fault-tolerant algorithm for each consensus group. Each consensus group verifies only a portion of the transactions. The TPS and scalability can be increased linearly by increasing the consensus group. At the same time, the blockchain system can still maintain 1/3 fault tolerance. The MBFT algorithm can reduce the forking of the blockchain while ensuring high fault tolerance and consensus speed.

2) We design a consensus node selection algorithm. By using the verifiable random function (VRF) and threshold secret sharing algorithm, all nodes can jointly decide the consensus nodes and prevent malicious nodes from using mathematical methods to attempt to improve their probability of being elected. The safety and stability of the blockchain system are guaranteed.

3) We design a credit incentive mechanism for the consortium blockchain. In a consortium blockchain, it is often not suitable to use tokens or coins as an economic incentive. We promote the stability of the network from the credit dimension through a credit incentive mechanism linked to consensus node election and network credit.

In this paper, we introduce blockchain technology and analyze the consensus algorithm in Section II. Then, we introduce the details of the consensus node selection algorithm and MBFT algorithm in Section III. Then, we analyze the scalability, safety, liveness and performance in Section IV. Finally, we summarize this paper in Section V.

II. BACKGROUND

A. THE ORIGIN OF BLOCKCHAIN

With the development of cryptography and Internet technologies, people are beginning to learn about cryptocurrencies. As early as the 1980s, the cypherpunk movement had presented the idea of cryptocurrencies. Timothy May proposed a non-traceable e-money, Crypto Credits, to reward hackers who are committed to protecting citizens' privacy. In 1990, David Chaum proposed Ecash, a non-traceable cryptographic network payment system based on blind signature technology. In 1998, Dai Wei proposed an anonymous and distributed electronic cryptocurrency system: B-money. In 2005, Nick Szabo proposed the idea of BitGold [16].

In 2008, Nakamoto released the Bitcoin White Paper (Bitcoin: A peer-to-peer Electronic Cash System). This paper expounds his new conception of e-money and designs the Bitcoin based on the bottom of the blockchain. Bitcoin uses the PoW consensus algorithm to solve these problems by introducing the gaming theory.

B. DEVELOPMENT OF CONSENSUS ALGORITHMS IN BLOCKCHAIN

PoW was the earliest consensus algorithm used in a blockchain. To ensure security and reduce forking, 10-minute block time and a 1-MB block size were established. Moreover, to prevent double-spend attacks, transactions on a block still need to wait for the confirmation of more than six subsequent blocks. This results in Bitcoin providing only 7 TPS. The TPS of ETH is in the tens of TPS, which cannot meet the needs of high-frequency trading systems. To solve these problems, many researchers have attempted to improve the PoW mechanism. As an example, Ittay Eyal *et al.* proposed a new consensus algorithm: Bitcoin-NG [13].

In Bitcoin-NG, miners still use PoW to compete with hashing power. The winning miner collects transactions in

the current blockchain network, validates them and publishes them in micro blocks until the next new block is generated by another miner. In this way, the number of additional blocks is increased while keeping the hashing power competitiveness unchanged. Transaction processing speed is simply determined by the computing power of the miner currently generating the block.

Based on Bitcoin-NG, Byzcoin added joint signature and PBFT to make further improvements [11]. In Byzcoin, the window size w (the number of consensus nodes) needs to be selected first, and then, the w miners that generated the most recent blocks are selected as members of the consensus group. The window moves forward with the emergence of new miners, and the total number of members in the consensus group will keep w unchanged. The latest miners are the leader nodes of the consensus group. Then, the PBFT algorithm is executed in the consensus group to generate micro-blocks at a certain speed until the next mining block is generated and a new leader node appears.

Sunny King *et al.* first realized proof of stake (PoS) in PPcoin (PPC), issued in August 2012 [17]. In PoS, digital money has a currency age page, which is equal to the number of coin holdings × holding time. The longer that each node holds the currency, the more rights it has in the network. Simultaneously, the holder of the currency will obtain a certain income according to the age of the currency. This mechanism encourages miners to increase the holding time of their coins. Here, proof on the blockchain no longer depends entirely on workload, effectively solving the problem of resource waste under PoW mechanisms. Moreover, in the PoS mechanism, the security increases with the total value of the blockchain network. Attacks on this blockchain require attackers to hold a large number of digital currencies for a long time. The costs of attacks have increased considerably.

Compared with PoW, the PoS consensus algorithm is energy efficient and has both high efficiency and decentralized computing power. However, such an algorithm is also more easily forked and exposed to vulnerabilities of long-range attacks and Nothing at Stake attacks [18]. Many researchers have also focused on improving the PoS algorithm. Algorand [19] and Ouroboros [20] are both improved algorithms that are based on PoS and use VRF. In Algorand, all eligible users can participate in an encrypted lottery. The user's account balance determines the probability of being selected as a block producer. All lottery users form a block consensus group. The nodes in the block consensus group then apply PBFT-like algorithms to determine the final block to be generated. In Ouroboros, all eligible nodes have a chance to become block producers in the next stage. The nodes publish encrypted random numbers in a specific stage and then decrypt and publish the random numbers in the verification stage. Then, VRFs are used to randomly select the consensus nodes of each block in the next stage from these nodes. These nodes generate blocks in a determined order. Algorand and Ouroboros, which combine PoS, VRFs, Secure Multi-Party Computation (SMPC) [21], PBFT and other improved algorithms, can effectively reduce system forking and improve the speed of consensus, which is the main direction of current academic research.

Bitshares proposed a Delegated Proof of Stake (DPoS) technique in 2013 [22]. Each miner can vote for a representative according to their share rights and interests. The top nodes of the network that participated in the election and obtained the most votes obtain the right to package blocks. They package these blocks in a predetermined order and thus obtain a certain reward. A representative node of a successful election needs to pay a certain amount of tokens and must guarantee online time. DPoS combines the characteristics of PoW and PoS. All nodes can choose their own voting objects, which reduces computing resource utilization and improves efficiency. Moreover, each participating node has the right to vote. When there are sufficient nodes in the network, the security and decentralization of DPoS can be guaranteed.

Some blockchains also use new data structures, e.g., the directed acyclic graph (DAG) used by the Greedy Heaviest Observed Subtree GHOST [23] and IOTA. The DAG skips block packaging by directly linking each transaction. This improves throughput through high concurrent transaction confirmation. In addition, there are algorithms that choose to replace the PoS method with other decentralized mechanisms such as the Proof of Space consensus mechanism and the Proof of Elapsed Time (POET) consensus mechanism [24].

C. THRESHOLD SECRET SHARING ALGORITHM

In 1979, Shamir [25] and Blakley *et al.* [26] gave the earliest threshold secret sharing algorithm using algebra and geometry, respectively. The Shamir threshold secret sharing scheme divides the secret information S into n pieces of sub-secret information $S1, S2, \ldots, Sn$ and distributes the sub-secret to n participants, and k or more sub-secrets can be used to reconstruct the secret information S. Any sub-secret of $k-1$ or less cannot obtain any information of S, where $k \leq n$. The Shamir threshold secret sharing scheme assumes that both the distributor and the sharer are honest; the verifiable secret sharing scheme removes this assumption and therefore has greater security in practical applications. The two most famous verifiable secret sharing schemes are the first non-interactive (t, n) threshold verifiable secret sharing scheme (Feldman-VSS) proposed by Feldman in 1987 [27], which did not require a trusted authority, and the first information-based secure non-interactive verifiable secret sharing scheme (Pedersen-VSS) later proposed by Pedersen.

III. DESIGN OF MBFT CONSENSUS ALGORITHM

There is a large difference between consortium blockchains and public blockchains. A public blockchain runs across the Internet. All nodes can access a public blockchain, and all the information on the blockchain is publicly shared. The consortium blockchain mainly runs in a relatively closed environment. The nodes have different roles and functions. Some nodes with high trustworthiness are approved and are responsible for verifying transactions and packaging the blocks. The

remaining nodes are the synchronization node, which is only responsible for synchronizing the existing block information; the client is only connected to the synchronization node. Among the consortium blockchains, IBM's main open-source project, Hyperledger Fabric [28], is a representative open-source consortium blockchain and is structurally consistent with the typical consortium blockchain architecture. In version 0.6 of Hyperledger Fabric, the functions of authorization and endorsement are integrated into the consensus node. All nodes are consensus nodes. This design leads to a heavy burden on the nodes and has a large impact on the TPS. After version 1.0, the functions of the nodes were separated. The nodes are divided into endorsers, orderers, and committers, which separate the functions of the nodes and improve the efficiency of the consensus. However, the consensus algorithm that it uses is based on Kafka and cannot tolerate Byzantine mistakes. Kafka can only address crash faults; it cannot withstand Byzantine faults.

To improve the TPS of the consortium blockchain and to consider fault tolerance, this paper proposes a new consensus algorithm - Mixed Byzantine fault tolerant (MBFT). The nodes are divided into verifying nodes, backup nodes, and clients. The nodes in the blockchain maintain a verifying node list. The verifying node list contains the public keys of all the verifying nodes. The node identifies the identity of the verifying node through the public key in the verifying node list. The consensus algorithm is the core of the blockchain, which affects the security and efficiency of the blockchain. This chapter introduces the consensus group, transaction verification process, and incentive mechanism in MBFT.

A. DETAILS OF MBFT

Traditional public blockchains, such as Bitcoin and Ethereum [29], need to address complicated network environments. Nodes do not have a trust relationship with each other, and the network communication environment is unstable. Thus, most public blockchains use a final consistency consensus algorithm, such as PoW or PoS, to ensure security through gamification. In the context of the consortium blockchain, the above problems have been addressed to an extent. For example, there are "weak trusts" between nodes in the consortium blockchain. In the consortium blockchain, some nodes authenticated by enterprise credit endorsements or asset mortgages are responsible for the verification and consensus of transactions.

The overall structure of the blockchain is shown in Fig. 1. The nodes that can participate in transaction verification are called verifying nodes. The verifying nodes are the core node of the blockchain and are responsible for verifying transactions in the entire blockchain and the packages of blocks. In the consortium blockchain architecture, any node that wants to be a verifying node needs to be confirmed by other nodes in the network. Most verifying nodes vote together to decide whether to accept a new verifying node. The backup node is the candidate of the verifying node. When electing the verifying node, the backup node can also participate in the election and become a new verifying node. The backup node is responsible for verifying the block packaged by the verifying node and both checking and reporting malicious behavior by a verification node. Other nodes that do not participate in transaction verification but can initiate transactions are clients.

B. CONSENSUS IN LCG

The verifying nodes are classified into two layers: the low-level consensus group (LCG) and the high-level consensus group (HCG). Every transaction is sent to a backup node and then allocated to an LCG. The communication cost is O(n). When the consensus of LCG is reached, the primary nodes in the LCGs will package the transactions and send the mini-block to HCG. The nodes collect and verify the mini-blocks from different LCGs and package a large block. The detailed steps of the consensus in LCG are as follows:

Step 1. The client initiates a transaction request $\langle REQUEST, tx, t_c\rangle_{\sigma_c}$ to the backup node, where tx is the transaction requested to be executed, t_c is the timestamp of the client initiating the transaction, and σ_c indicates that client c signs the request.

Step 2. After receiving the request from the client, the backup node verifies the identity of the client and the timestamp t_c on the blockchain. If the authentication is successful and the time difference Δt between the transaction request timestamp and the latest block is less than the predefined time, the backup node generates the new request message $\langle REQUEST, tx, t_c\rangle_{\sigma bi}$ and sends it to the primary node in the low-level consensus group determined by the transaction allocation rule, where σ_{bi} is the signature of the backup node i to the client request. The transaction allocation rule is to perform the modulo operation on the first input of each transaction, and the result is the assigned LCG number.

Step 3. The node in an LCG forwards the request to the primary node in the group. The primary node needs to verify the transaction. It confirms whether the signature of the backup node is correct and whether the transaction conflicts with other transactions in the transaction waiting pool $Epool_p$ or packaging pool $Ppool_{lni}$ or that has been recorded in the blockchain. If verification is successful, the transaction is numbered and signed as $\langle TRANSACTION, \langle REQUEST, tx, t_c\rangle_{\sigma bi}, cycle, m_{\sigma p}$, where $cycle$ is the current $cycle$ of the LCG, m is the number of transactions from the current primary node, and σ_p is the signature of the primary node p.

Step 4. The primary node in the LCG then broadcasts the message to all nodes in the same consensus group. When the node (ln_i) in the LCG receives the transaction message, it first verifies the number of the transaction and the signature. After confirming the information, the node adds the transaction to the local transaction waiting pool $Epool_{lni}$ and sends $\langle AGREE, cycle, m\rangle_{\sigma_{lni}}$ to the primary nodes in the consensus group. All the node stores received $AGREE$ messages.

Step 5. If the primary node ln_i receives $2f+1$ $AGREE$ messages for a certain transaction, the transaction is moved out of

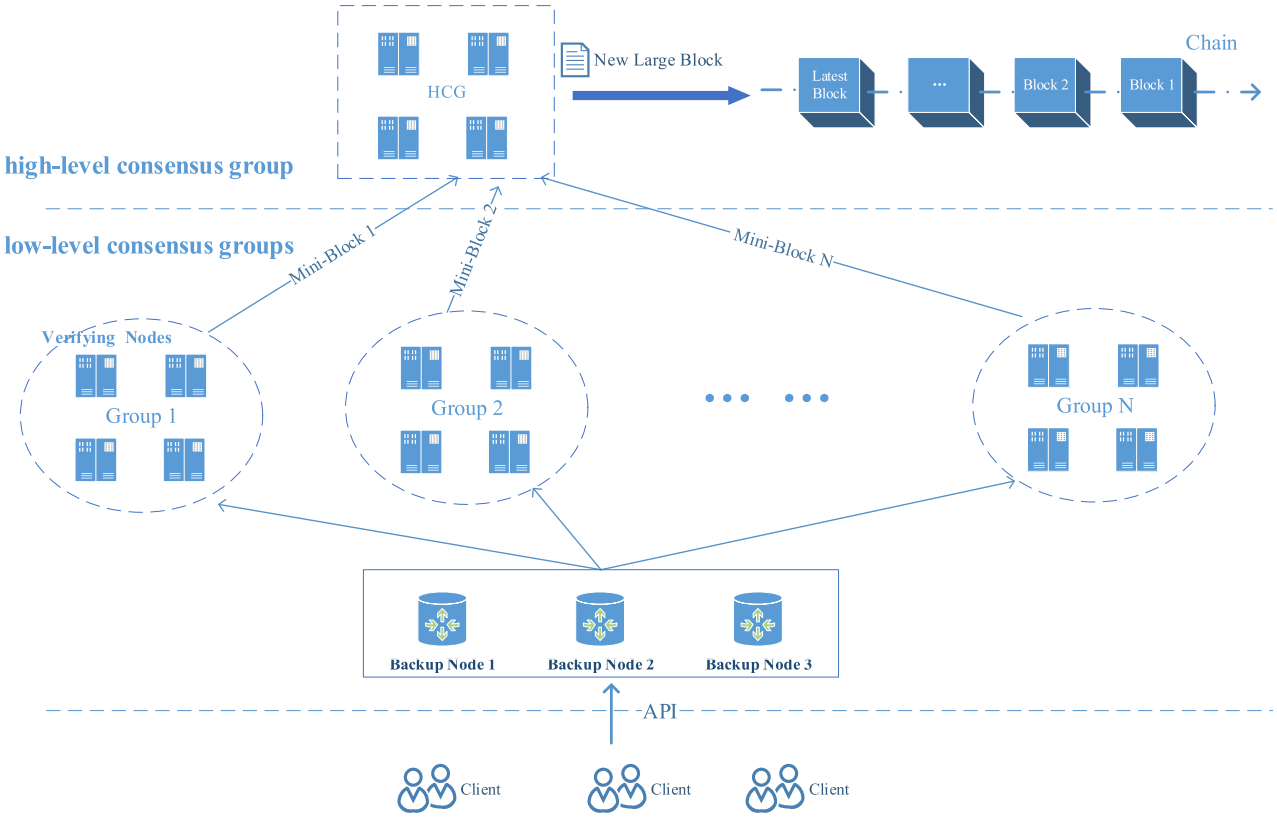


FIGURE 1. 2-layer Structure of the MBFT.

the local transaction waiting pool $Epool_p$, and the transaction, together with the received $2f+1$ *AGREE* messages, is put into the transaction packaging pool $Ppool_p$.

Step 6. t_1 second (the preset packaging time) after receiving the newest large block, the primary node p in the consensus group will package the transaction in the transaction packaging pool $Ppool_{lni}$ and send the block information $\langle MINIBLOCK, height, mini\_block_{lcg}\rangle_{\sigma_p}$ to the other nodes in the same consensus group, where h is the height of the current block, and *mini_block* is the block that has been verified and packed by the node.

Step 7. After receiving the block information packed by the primary node, node i in the same consensus group performs verification on the block information and each transaction included in the block. The node needs to confirm that the transactions in the mini-block are proper or in the local $Epool_{lni}$. After verification, the node broadcasts $\langle AGREE\_BLOCK, HASH\ (mini\_block_{lcg}), height\rangle_{\sigma i}$ information to other nodes in the same LCG. When the primary node receives sufficient *AGREE_BLOCK* information, it can send $\langle AGREE\_BLOCK_{LCG}, mini\_block_{lcg}, [sig_i(mini\_block_{lcg})], height\rangle_{\sigma p}$ to the high-level consensus group, where $block_{lcg}$ is the block obtained by LCG after consensus on the current block height, and $[sig_i\ (mini\_block_{lcg})]$ is the collection of signatures in the LCG; these signatures are from the *AGREE_BLOCK*. The transaction can continue to be verified and pushed into the $Ppool_{lni}$ after sending out the mini-block.

Because the LCG only needs to maintain the final consistency after the large block is recorded on the blockchain. Therefore, each verifying node in the consensus group may have transactions in *Epool* or *Ppool* after packaging a mini-block. If certain conditions are met (for example, the transaction's set time is not exceeded), these transactions can be confirmed preferentially in the new round of consensus. Therefore, the unconfirmed transactions in the previous round of consensus are handled as follows.

After receiving the large block, all nodes i verify the large block and retrieve the transaction in the local $Epool_i$ and the $Ppool_i$ according to the large block and remove transactions that have been packaged into the large block. In the new round of consensus, the node will address the existing transactions in the local pool and send the transactions that are in the $Epool_i$ and that satisfy the transaction waiting time requirement but that are not packaged into the block in this round to the primary node in the consensus group.

C. CONSENSUS IN HCG

After receiving the mini-block sent by the LCGs, the nodes in the high-level consensus group need to check, for the mini-block,

1) whether the signatures in each mini-block are sufficient, 2) whether all the signatures of verifying nodes are correct, 3) whether the hashing of the previous block points to the current large block, and 4) whether the input of all transactions is conflicted. The conflicting transactions will be marked and

recorded in the special conflicting transaction zone in the large block. After the verification is successful, the mini-blocks are arranged sequentially. After a certain time or after receiving all the mini-blocks, a large block will be packaged and added to the blockchain. The detailed steps for consensus in the HCG are as follows:

Step 1. After completing the above-mentioned verification, the nodes in the high-order consensus group return the confirmation message $\langle RECEIVE_{BLOCK}, HASH(mini\_block_{lcg}), height\rangle_{\sigma i}$.

Step 2. The nodes in the LCG are responsible for confirming that the mini-block is received by most of the nodes $(2f+1)$ in the high-level consensus group.

Step 3. If the primary node in the HCG collected all the mini-blocks or more than 3/4 mini-blocks in a max packaging time, it broadcasts the sequence number confirmation information of the mini-blocks $\langle BLOCK, HASH(mini\_block_{lcg}), height\rangle_{\sigma i}$ and collects the agree messages from the nodes in the high-level consensus group, where the *height* is the number of current large blocks and σ_i is the signature of the primary node.

Step 4. The node i in the high-level consensus group verifies whether the sequence issued by the primary node is legal. If the nodes agree with the sequence issued by the primary node, the node returns the AGREE message. if certain mini-blocks $mini\_block_{lcg}$ are missing locally, the node requests the block information from the corresponding nodes and returns $\langle AGREE, [HASH(mini\_block_{lcg})], height\rangle_{\sigma_i}$ after the verification is successful. If the verification ultimately fails, it broadcasts $\langle DISAGREE, [HASH(mini\_block_{lcg})], height\rangle_{\sigma_i}$. If any node collects $2f+1\langle DISAGREE\rangle$ messages, it can broadcast the $[\langle DISAGREE\rangle]$ set and enter the next block stage.

D. SELECTION OF THE CONSENSUS NODES

With sharding technology, the number of nodes that need to be controlled against malicious attacks is reduced. Moreover, because the verifying node can obtain a certain credit boost, a mechanism is needed to determine the node allocation. Compared with the public blockchain, the consensus nodes in the consensus of the consortium blockchain are limited and known. In this case, the consensus nodes in consortium blockchain can interact with each other through the known node list, which can effectively prevent the Sybil attack. Moreover, we can combine the threshold secret sharing algorithm with random number generation directly to construct an interactive real random number generation scheme on the blockchain. Therefore, we design a random number generation method on the blockchain-based on the VRF and threshold secret sharing algorithm. Then we design a consensus node election algorithm based on the random number generation method and the node's credit score system.

In MBFT, each 5000-block times is called a ***cycle***. In one cycle, the consensus group is determined in the previous cycle. The end of the cycle is the election period. The election period lasts for $3x$ blocks, with the first x blocks being the commitment period, the next 100 blocks being the disclosure period, and the last 100 blocks being an extra disclosure period. All the nodes that want to become verifying nodes in the next cycle need to generate a random number and calculate a hash value. Then, each node sends an election transaction $\langle ELECTION, Hash(sig_i(R_i)), sig_i(R_i), [E_{\rho k}\{(R_{ik})_{\sigma i}\}]\rangle_{\sigma i}$ to the blockchain, where R_i is the random number of nodes i and R_{ik} is the sub-random number that is generated with a (t, n) threshold verifiable secret sharing scheme. t should be $f+1$, and n should be greater than $2f+1$. In this way, the malicious nodes cannot obtain the random number, and the good verifying nodes can jointly disclose the random number in the next period.

During the disclosure period, all consensus nodes need to disclose the random number published during the commitment period. In the extra disclosure period, all the verifying nodes retrieve the transactions to identify the unpublished random numbers in the disclosure period. If a sub-random number is encrypted with a verifying node's public key, the node must publish the corresponding sub-random number after decryption during this period. Then, if more than $f+1$ good verifying nodes publish the sub-random number, the original random number can be restored. In addition, we introduce computational evidence to the last block of the public period to prevent collusion between nodes in the high-level consensus group.

All nodes can obtain the public keys and all random numbers of all the nodes participating in the election on the blockchain. Then, any node can calculate the same $\mathrm{R} = R_1 \& R_2 \& \cdots \& R_n$ locally, where & is a bitwise AND operator, and R is a true random number generated by all the selected nodes. Then, the distance $D_i = W_i/K \cdot abs(R-\rho_i)$ between the candidate node i and the random number R can be calculated, where W_i is the credit weight of node i, K is the highest credit score and ρ_i is the public key of node i. Because R_i and node public keys are recorded on the blockchain, the results calculated by all nodes are consistent. Then, all the nodes select the same verifying nodes with the smallest distance. Next, in order, the verifying nodes are assigned to the high-level consensus group and the low-level consensus group according to the number of nodes in each consensus group and the allocation rules. All other candidate nodes are backup nodes. Thus, at the beginning of the new cycle, all nodes can obtain a consistent node assignment locally by retrieving information on the blockchain.

During the election of the verifying nodes, there may be cases whereby some malicious nodes collude to manipulate the election results. For example, during the disclosure period, the malicious nodes may calculate the current random number seed according to the random number already published by other nodes and then selectively publish the random number of some malicious nodes, thereby manipulating the ultimately generated random number and affecting the outcome of the election. To this end, we designed an extra disclosure period during which we disclose random numbers that were not published by the original node during the

disclosure period for various reasons. Therefore, the design of the random selection algorithm needs to satisfy three requirements: 1) In the commitment period, a random number of one node should be divided into multiple sub-random numbers and saved by other nodes; however, any node using fewer than $f + 1$ sub-random numbers cannot restore the random number in advance. 2) In the extra disclosure period, a node can use a certain number of sub-random numbers to restore the original random number. Thus, for the random election of the consensus group, we chose Feldman-VSS to distribute random numbers in the commitment period. The main steps of Feldman-VSS are as follows:

p is a large prime number, $p \gg n > t$, and p needs to be larger than the maximum value of the x node's number R_x.

The node distributes its sub-secrets to other nodes:

1. Select the random number $a_1, \ldots, a_{t-1}$ on $GF(p)$ to construct a polynomial

$$f(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1}$$

where $R_x = a_0 = f(0)$

2. Calculate $A_i = g^{a_i} \bmod p$, $i = 1, 2, \ldots, n$ and broadcast A_i

3. Calculate $r_i = f(i) \bmod p$, $i = 1, 2, \ldots, n$ and send r_i to the node P_i.

When the r_i is received, the node P_i verifies whether the following equation holds:

$$g^{r_i} = \prod_{j=0}^{t-1} A_j^{i^j} \bmod p$$

If the equation holds, r_i is valid; otherwise, it is invalid.

R_x recovery:

Suppose that a group of nodes $P_1, \ldots, P_k (k \geq t)$ cooperates to recover the R_x, that is, $r_1, \ldots, r_k$ are known. The nodes first check whether the r_i submitted by other nodes are valid and exclude the invalid shares. When there are at least t valid shares, they use the Lagrange interpolation formula to calculate $f(x)$,

$$f(x) = \sum_{i=1}^{t} r_i \prod_{j=1, j\neq i}^{t} \frac{x-i}{j-i} \bmod p$$

thereby recovering the secret s as

$$R_x = f(0) = \sum_{i=1}^{t} r_i \prod_{j=1, j\neq i}^{t} \frac{i}{i-j} \bmod p$$

E. NODE CHANGE

In PBFT, considering the consistency of the recorded information of each node, a three-phase consensus is adopted. In MBFT, we redesigned the global view change because of the characteristics of blockchain. When packaging the mini-block, each verifying node needs to link the latest large block as the previous block. Only when the link is correct can the mini-block be packaged into the large block. This solves the problem of transaction consistency. In the consensus group of MBFT, when the primary node cannot respond or have package error, it will cause the ''primary node change''. If the node change is triggered in the consensus group, each node can become the primary node in turn according to the order of the distance between the node and the random number. When the new node becomes the primary node, it sends the latest large block that it plans to link to. After 2/3 nodes' consent information is collected, the primary node broadcasts the collected confirmation information, and the consensus group begins to work. The following are the conditions that cause primary node change in the consensus group.

1. When Δ_n consecutive mini-blocks submitted by an LCG are not packaged into the large blocks, the LCG will change the primary node and record this primary node as ''dereliction''. If $\Delta_n * n$ consecutive large blocks cannot contain the mini-block from an LCG, the LCG will be removed in this cycle, where n is the number of nodes in LCG.

2. If the HCG fails to package large blocks within the specified time Δ_t, the HCG will change the primary node and record the ''dereliction'' nodes in the new large block. If the large blocks cannot be packaged in $\Delta_t * m$ time, the next LCG will become HCG, where m is the number of nodes in HCG.

3. When there is a transaction in the mini-block that does not belong to the consensus group, the nodes in this consensus group signed for this transaction will be recorded as ''dereliction'', and this LCG will change the primary node.

4. If in an LCG more than x transactions signed by more than 2/3 nodes have been collected by a node, but the primary node still submits empty mini-blocks to the HCG. This LCG will change the primary node.

5. If in the HCG more than x mini-blocks signed by more than 2/3 nodes have been collected by a node, but the primary node still submits the empty large blocks, the HCG will change the primary node.

6. If Δ_m consecutive large blocks cannot contain more than 1/2 mini-blocks, the HCG will change the primary node. If $\Delta_m * m$ consecutive large blocks cannot contain more than 1/2 mini-blocks the next LCG will become HCG, where m is the number of nodes in HCG.

To speed up the recovery of the system, if a node finds that the HCG is incorrect in $\Delta_m * m * k$ blocks or $\Delta_t * m * k$ time, it can initiate a cycle change proposal in its consensus group, where k is a variable related to the number of consensus groups. If more than 2/3 of the nodes in this group agree, the group initiates a re-election proposal message and sends it to all the verifying nodes. If any verifying node receives the re-election proposal message and confirms that the large block is incorrect, it must reply with an agree message. When the group has collected more than 2/3 of the agree messages from all the verifying nodes, it can replace the previous HCG and generate new large blocks for re-election. The blockchain enters the election cycle. The new HCG is responsible for the block of this election period.

F. INCENTIVE MECHANISM

We have added a credit sub-mechanism to the consensus algorithm to score the verifying nodes based on the contribution to

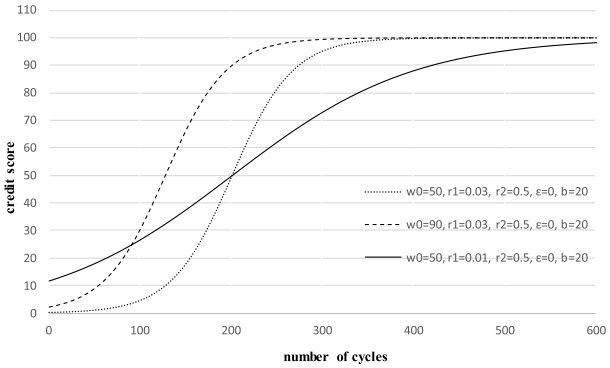


FIGURE 2. Score Curves of Different Coefficients.

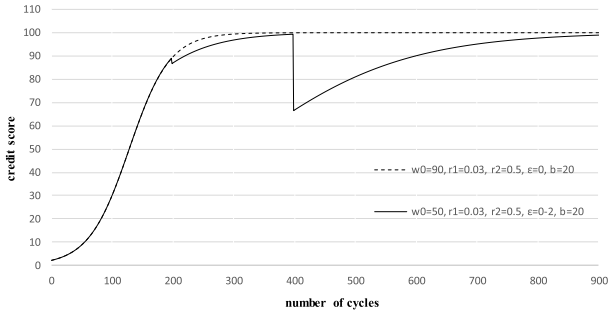


FIGURE 3. Credit Score Curves When a Node Obtains a "Dereliction" At its Cycles of 200 and 400.

the blockchain network. If the consensus group is not able to verify the transaction and package the community according to the system rules, the corresponding credit will be deducted. The credit score is based on real-time data in the blockchain, and the credit of the node can be dynamically evaluated in real-time by each node according to pre-established rules. The formula of credit evaluation is as follows:

$$W = \frac{Kw_0 e^{r_1 r_2^{\varepsilon}(t_c - 200 - b^{\varepsilon})}}{K + w_0(e^{r_1 r_2^{\varepsilon}(t_c - 200 - b^{\varepsilon})} - 1)}$$

K is the maximum value of the function, w_0 is the starting value when a node joins the system, r_1 is the growth coefficient, r_2 is the penalty coefficient, ε is the number of "derelictions" of the node (for example, a node in HCG signed a double-spending transaction), t_c is the number of cycles that the node participates in as a verifying node, and b is the regression coefficient. All variables can be obtained on the blockchain, so each node can use the same data to calculate the consistent credit score of all nodes. The coefficient can be set when the blockchain system is initialized as required.

This credit evaluation formula is based on a sigmoid function. Fig. 2 shows the function curve of credit score when selecting different coefficients. When nodes participate in the blockchain, they all go through a period in which the credit score increases slowly. Then, with the increase in the number of verifications, the final credit value of the node will increase and tend to be constant. The change of the node credit score is in a certain range, which can prevent the occurrence of a "supernode". Due to the introduction of the penalty mechanism, when the node produces "dereliction", the current credit score will be greatly reduced, and then its credit score growth rate will be slower. The penalty effect will be accumulated exponentially, which can quickly eliminate malicious nodes and increase their cost. Fig. 3 shows the credit score assuming that the node obtains a "dereliction" at its cycles of 200 and 400.

In later research, we will try to add the credit score of clients according to the application scenarios. For example, if the low-level consensus group or the node detects a double-spending attack initiated by the client, the node in the consensus group can obtain a credit bonus. The client would reduce the credit score. If the client credit score is too low, its asset will be frozen, and the transaction will be prohibited.

IV. ANALYSIS OF THE MBFT

A. SCALABILITY

Our designed blockchain uses a two-layer structure and sharding technology. The sharding technology changes the way in which all nodes jointly verify a transaction in a traditional consensus algorithm. In our blockchain, only a subset of the nodes is needed to verify each transaction. Then, the transaction is packaged into a mini-block and verified by the HCG. A node does not need to verify all the transactions as in traditional consensus algorithms. Suppose there are k LCGs, the communication complexity of all the LCGs is $O(n/k)$. The number of nodes in HCG is much smaller than that of LCGs. When the number of nodes is large enough, the communication complexity of HCG can be regarded as constant. Thus, the communication complexity of MBFT is $O(n/k) + O(k)$. As the number of nodes in the consensus group increases, the TPS will grow linearly, and the communication traffic will only grow linearly as well.

B. SAFETY AND LIVENESS

1) DOUBLE SPENDING AND CHAIN FORK

A "double spending" problem may be encountered in transactions between mini-blocks. For example, a malicious client may generate two transactions that have the same input address. Both transactions will be packaged into different mini-blocks. However, in MBFT, if this occurs in two mini-blocks, the node will put the two conflicting transactions into the conflict pool at the same time and form a conflict pair. Finally, when the large block is generated, the conflict pool is added to the large block. The two transactions are invalidated, and the amount of the conflicting address is awarded to the verifying nodes. The initiator of double-spending attack will not obtain any benefits.

The proportion of malicious nodes is less than 1/3 of all nodes, and the consensus of 2/3 nodes in an LCG is needed for mini-block packaging. Therefore, malicious nodes can control no more than 1/2 of LCGs. Each large block needs to contain more than 3/4 of the mini-blocks, so there will not be two large blocks with the same height.

TABLE 1. Comparison of consensus algorithms.

	PoW	*PoS*	*DPoS*	*Elastico*	*OmniLedger*	*Monoxide*	*Kafka*	*PBFT*	*MBFT*
Scalability	Strong	Strong	Weak	Strong	Strong	Strong	Strong	Weak	Strong
Energy saving	No	Partial	Partial	Partial	Partial	No	Yes	Yes	Yes
Block fork	Yes	Yes	No	No	No	Yes	No	No	No
Byzantine fault tolerance	50%	50%	50%	33%	33%	50%	No	33%	33%
Scalability in Fragmentation	no	no	no	Weak	Weak	Strong	Weak	no	Strong
Node management	Public	Public	Public	Public	Public	Public	Permissioned	Permissioned	Permissioned
Transaction verifying delay	High	High	High	High	Low	High	Low	Low	Low
TPS	10	10^2	10^2-10^3	10^2 -10^3	10^3- 10^4	10^2 -10^3	10^3- 10^4	10^2 -10^3	10^3- 10^4

2) ATTACK IN THE LOW-LEVEL CONSENSUS GROUP

When the proportion of malicious nodes is less than 1/3 in a single consensus group, the generation of correct blocks is not affected by this consensus group. In contrast, if the proportion of malicious nodes in a single consensus group exceeds 1/3 but is less than 2/3, due to the algorithm's fault tolerance mechanism, the consensus group will not package a qualified mini-block and it will be removed in this cycle after a period of time.

When the proportion of malicious nodes in an LCG exceeds 2/3, the malicious nodes in this LCG can package an incorrect mini-block. In this scenario, if the HCG can still produce the correct large block, the incorrect mini-block can be identified and not be included in large blocks and this LCG will be removed in this cycle after a period of time. Transactions will be reallocated to the remaining LCGs. If more than 1/2 of the LCGs are unable to package the legal mini-blocks, they will be re-elected after a period of time

3) ATTACK IN HIGH-LEVEL CONSENSUS GROUP

If the percentage of malicious nodes in the HCG exceeds 1/3, the large block will be absent or incorrect. In the worst-case scenario, the percentage of malicious nodes in most LCGs also exceeds 1/3. However, according to drawer principle, at least one loyal consensus group exists. As described before, this loyal group can replace the previous HCG or, initialize a correct large block and collect more than 2/3 of the verifying nodes' votes to begin a new election period.

In summary, the use of a 2-layer structure and sharding technology greatly improves both the scalability and the TPS. Moreover, by designing the cycle change to switch from the sharding state to the global state, the fault tolerance of the blockchain is still 1/3.

C. PERFORMANCE

We evaluated the MBFT's performance on 60 servers. Each server was equipped with an E5-2620 2.1 GHz CPU and 16 GB of RAM. The block packaging time is 3 s, and the size of the mini-block is 1 M. Each LCG and HCG contains 4 nodes. There are 5 clients and 1 HCG. We varied the number of LCGs from two to ten, performed five experiments for each combination, and recorded the average throughput. Fig. 4 shows the TPS test results, querying latency and confirmation latency. The query latency is the time required for a client to obtain a transaction from the blockchain. The confirmation latency is the time between the client sending a transaction and the time of a transaction being recorded on the blockchain.

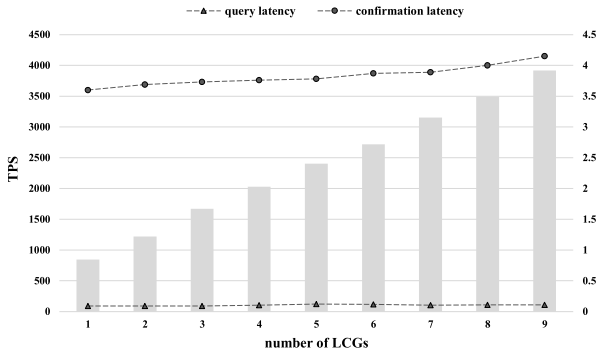


FIGURE 4. TPS and Latency of MBFT.

From the experimental results, MBFT has good scalability. The TPS increases proportionally as the number of LCGs increases. In addition, the confirmation latency does not increase significantly when the number of LCGs increases. The TPS of MBFT with 10 LCGs is about 4000. This performance level is sufficient to satisfy the requirements of actual production environments.

D. COMPARISONS

We list the nine consensus algorithms in Table 1. PoW, PoS and DPoS are usually used in public blockchains. They have good Byzantine fault tolerance but low TPS. They also require substantial computing resources, and they guarantee the security of the blockchain system through com-

puting power. Hence, they are not suitable for consortium blockchain.

The Byzantine Fault Tolerance (PBFT) algorithm was the first consensus algorithm used in consortium blockchains. MBFT originates from PBFT, but they differ in many ways. The consensus nodes in PBFT are all of the same levels. Thus, every node needs to communicate with all the other nodes to verify each transaction. Consequently, when the number of nodes increases, the number of communications grows exponentially, making its scalability poor. The Hyperledger fabric 1.0+ uses Kafka to reach a consensus. However, Kafka can only address crash faults; it cannot withstand Byzantine faults.

Elastico [10] is the first blockchain that applied sharding technology. In Elastico some verifying nodes are selected through a PoW algorithm. Then these verifying nodes are assigned to different committees. The committees execute PBFT algorithm and generate transaction packages. After more than two-thirds of the nodes sign the transaction package, the transaction package is submitted to the consensus committee. The consensus committee verifies the signatures and packs all the transactions into blocks that are recorded on the blockchain. Elastico verifies the availability of sharding technology on the blockchain. On a certain scale, the sharding technology can expand the throughput almost linearly. However, Elastico uses PoW to select the consensus nodes, which consumes extensive time and makes the transaction delay very high. Moreover, the PBFT algorithm used in each committee has a high communication complexity. When the number of consensus nodes in each committee is large, the delay is also high.

On the basis of Elastico, OmniLedger uses a bias-resistant public-randomness protocol instead of PoW to select the consensus nodes and then uses RandHound protocol to classify these consensus nodes into different segments [14].To solve the atomic transactions across different segments, OmniLedger introduces the Atomix protocol. However, PBFT is still used as the consensus algorithm in each segment and cross sharding transactions are considered. The communication complexity is high.

Wang et. al proposed Monoxide in 2019 [30]. Monoxide introduces sharding technology into the PoW blockchain system and improves the TPS of PoW. In addition, the Chu ko-nu mining algorithm is used by Monoxide to solve the problem of hashing power dispersion caused by sharding, so that each miner can mine in different segments at the same time.

Compared with these algorithms, MBFT uses a two-layer design and sharding technology, which substantially reduces the communication between nodes. Thus, it achieves faster consensus and better scalability. MBFT designs a random node selection algorithm based on VRF and Feldman VSS to select the consensus nodes. In the normal operation of the system, the election phase overlaps with the phase of the normal packaging block without additional time. At the same time, combined with the characteristics of controllable verifying nodes in consortium blockchain, the global re-election mechanism and credit evaluation mechanism are designed. By optimizing the global node switching mechanism, two-phase Byzantine fault tolerance (BFT) consensus is used in fragmentation. It reduces the complexity, ensures the low delay and high TPS of the transaction, and also guarantees the scalability of the system in each sharding.

During the normal operation of the system, the election phase overlaps with the block packaging phase without additional time. In addition, MBFT combines the characteristics of node controllability in consortium blockchain, designs a global re-election mechanism and a credit evaluation mechanism, and uses two-phase BFT consensus in each segment. This reduces the communication complexity and ensures low latency and high TPS. At the same time, it also guarantees the scalability in each segment.

V. CONCLUSIONS

We process a two-layer consensus algorithm in this paper: MBFT. The MBFT algorithm combines layering technology and sharding technology. Layering can be used to separate the functions of nodes. By assigning the entire verification function and demodulation process to different nodes, layering can effectively reduce the load of individual nodes and improve consensus efficiency. Sharding can allocate transactions to different node groups. When the number of transactions increases significantly, the system can dynamically increase the number of nodes and shards, thereby improving the processing power and decreasing delay. Moreover, all nodes in the traditional blockchain must verify all transactions. In contrast, the verifying nodes in the MBFT are only responsible for a certain number of transactions. The throughput of the blockchain is positively correlated with the number of nodes, and the blockchain has strong scalability.

To ensure the fault tolerance of the system, we design a random election algorithm based on the VRF and threshold secret sharing algorithm. The blockchain system can ensure that each cycle's verifying nodes are randomly selected, which can prevent malicious nodes from colluding under a monopolistic behavior, thus ensuring the blockchain system's fault tolerance. Finally, to address the characteristics of a consortium blockchain that may not have mining incentives, we designed a credit score system. The credit score is not the digital currency and is closely related to the priority of the election and the release transaction. By verifying and supervising the behavior of the transaction, the node can obtain a credit score reward and will be punished when there is a behavior that harms the blockchain security. Through this mechanism, both the desire for the node's long-term participation and the cost of a malicious node attacking the system can be increased.

In the future, we plan to add zero-knowledge proof and secure multi-party computing solutions to increase the privacy protection of decentralized transactions. We also plan to expand MBFT into a framework of consensus algorithms, enabling users to customize the consensus algorithms used in

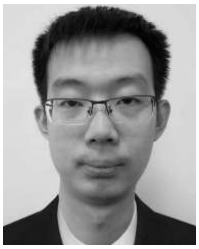
the consensus group while ensuring the fault tolerance of the entire blockchain system.

ACKNOWLEDGMENT

This research was supported by Suzhou Tongji Blockchain Research Institute. The authors thank the assistance of Xiuze Mao and Xianxian Zhou in system implementation and testing.

REFERENCES

[1] L. Lamport, "Paxos made simple," *Acm Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.
[2] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congress)*, Jun. 2017, pp. 557–564.
[3] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White Paper 1-9, 2009.
[5] Y. Yuan and F. Wang, "Blockchain: The state of the art and future trends," *Zidonghua Xuebao/Acta Automatica Sinica*, vol. 42, no. 4, pp. 481–494, 2016, doi: 10.16383/j.aas.2016.c160158.
[6] Y. Zhang and J. Wen, "The IoT electric business model: Using blockchain technology for the Internet of Things," *Peer-to-Peer Netw. Appl.*, vol. 10, no. 4, pp. 983–994, Jul. 2017.
[7] H. Liu, Y. Zhang, and T. Yang, "Blockchain-enabled security in electric vehicles cloud and edge computing," *IEEE Netw.*, vol. 32, no. 3, pp. 78–83, May 2018.
[8] J. M. Roman-Belmonte, H. De la Corte-Rodriguez, and E. C. Rodriguez-Merchan, "How blockchain technology can change medicine," *Postgraduate Med.*, vol. 130, no. 4, pp. 420–427, May 2018.
[9] P. Treleaven, R. Gendal Brown, and D. Yang, "Blockchain technology in finance," *Computer*, vol. 50, no. 9, pp. 14–17, 2017.
[10] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 17–30.
[11] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. Usenix Secur. Symp.*, 2016, pp. 279–296.
[12] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. Symp. Operating Syst. Design Implement.*, 1999, pp. 173–186.
[13] I. Eyal, A. E. Gencer, E. G. U. N. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement.* 2016, pp. 45–59.
[14] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Proc. IEEE Symp. Secur. Privacy*, May 2018, pp. 583–598.
[15] M. Du, Q. Chen, J. Chen, and X. Ma, "An optimized consortium blockchain for medical information sharing," *IEEE Trans. Eng. Manage.*, early access, Feb. 3, 2020, doi: 10.1109/TEM.2020.2966832.
[16] U. W. Chohan, "A history of bitcoin," White Paper 1-10, Sep. 2017.
[17] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," Tech. Rep., 2012, pp. 1–6.
[18] E. Deirmentzoglou, G. Papakyriakopoulos, and C. Patsakis, "A survey on long-range attacks for proof of stake protocols," *IEEE Access*, vol. 7, pp. 28712–28725, 2019.
[19] J. Chen and S. Micali, "Algorand: A secure and efficient distributed ledger," *Theor. Comput. Sci.*, vol. 777, pp. 155–183, Jul. 2019.
[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Advances in Cryptology—CRYPTO*. New York, NY, USA: Springer, 2017, pp. 357–388.
[21] A. Kulshrestha, A. Rampuria, M. Denton, and A. Sreenivas, "Cryptographically secure multiparty computation and distributed auctions using homomorphic encryption," *Cryptography*, vol. 1, no. 3, p. 25, 2017.
[22] D. Larimer, "Delegated proof-of-stake (DPOS)," Bitshare, Murska Sobota, Slovenia, White Paper, 2014.
[23] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2015, pp. 507–527.
[24] W. Zhao, S. Yang, and X. Luo, "On consensus in public blockchains," in *Proc. Int. Conf. Blockchain Technol. (ICBCT)*, 2019, pp. 1–5.
[25] A. Shamir, "How to share a secret," *Commun. Acm*, vol. 22, no. 11, pp. 612–613, 1979.
[26] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. Int. Workshop Manag. Requirements Knowl.*, 1979, pp. 313–318.
[27] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. 28th Annu. Symp. Found. Comput. Sci.*, 1987, pp. 427–438.
[28] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Proc. Workshop Distrib. Cryptocurrencies Consensus Ledgers*, 2016, p. 4.
[29] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.
[30] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proc. 16th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2019, pp. 95–112.



MINGXIAO DU received the B.S. degree in automation from Tongji University, Shanghai, China, in 2014, where he is currently pursuing the Ph.D. degree.

His current research interests include blockchain and consensus algorithm.

QIJUN CHEN (Senior Member, IEEE) received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 1987, the M.S. degree from Xi'an Jiaotong University, Xi'an, China, in 1990, and the Ph.D. degree from Tongji University, Shanghai, China, in 1999.

He is currently a Full Professor with the College of Electronics and Information Engineering, Tongji University. He has published over 100 articles in journals and conference proceedings. His research interests include robotics control, environmental perception and understanding of mobile robots, and bio-inspired control.

XIAOFENG MA received the B.S. degree from the Taiyuan University of Technology, Taiyuan, China, in 1997, the M.S. degree from Tongji University, Shanghai, China, in 2000, and the Ph.D. degree from Leiden University, Leiden, The Netherlands, in 2008.

He is currently an Associate Professor with the College of Electronics and Information Engineering, Tongji University. His research interests include blockchain and big data.