Improving Byzantine Fault Tolerance Based on Stake Evaluation and Consistent Hashing

Guangfu Wu

wuguangfu@126.com

Jiangxi University of Science and Technology

Xin Lai

Jiangxi University of Science and Technology

Daojing He

Harbin Institute of Technology

Sammy Chan

City University of Hong Kong

Xiaoyan Fu

Jiangxi University of Science and Technology

Research Article

Keywords: Byzantine Fault Tolerance, Equity Assessment, Consistent Hashing, Security, Reliability, Distributed Systems

Posted Date: November 2nd, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3466608/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Peer-to-Peer Networking and Applications on April 11th, 2024. See the published version at https://doi.org/10.1007/s12083-024-01700-3.

Improving Byzantine Fault Tolerance Based on Stake Evaluation and Consistent Hashing

Guangfu Wu[1*], Xin Lai[1], Daojing He[2], Sammy Chan[3], Xiaoyan Fu[1]

[1]Department of Information Engineering, Jiangxi University of Science and Technology, GanZhou, 341000, JiangXi, China.
[2]Department of Software Engineering, Harbin Institute of Technology, ShengZhen, 518000, GuangDong, China.
[3]Department of Electronic Engineering, City University of Hong Kong, Hong Kong, 999077, China.

*Corresponding author(s). E-mail(s): wuguangfu@126.com;
Contributing authors: laixin0908@outlook.com; djhe@sei.ecnu.edu.cn;
eeschan@cityu.edu.hk; 563408523@qq.com;

Abstract

In the context of distributed systems, Byzantine fault tolerance plays a critical role in ensuring the normal operation of the system, particularly when facing with malicious nodes. However, challenges remain in enhancing the security and reliability of Byzantine fault-tolerant systems. This paper addresses these challenges by improving a Byzantine fault-tolerant approach based on stake evaluation and improved consistency hashing. We propose a method that leverages node stakes to enhance system security and reliability by allocating different trust values. Additionally, we introduce improvements to the consistency hashing technique, enabling its effective operation in a Byzantine fault-tolerant environment. By introducing redundant nodes on the hash ring to mitigate the impact of malicious nodes, we enhance system fault tolerance and scalability. Experimental results demonstrate a significant improvement in system security and performance using this approach. These findings suggest that our method holds considerable potential for widespread application in the field of Byzantine fault tolerance, supporting the development of more reliable blockchain systems.

Keywords: Byzantine Fault Tolerance, Equity Assessment, Consistent Hashing, Security, Reliability, Distributed Systems

1 Introduction

Over time, blockchain technology [1] has experienced rapid development and found widespread applications in numerous domains, ranging from finance to supply chain management and healthcare [2–4]. As a form of distributed ledger technology, blockchain addresses the inherent issues of trust and reliability found in traditional centralized systems. For instance, considering global financial transactions, blockchain technology ensures transparency, fraud prevention, and reduces intermediaries between financial institutions.

However, ensuring consensus among various nodes in a distributed environment is a critical

task, particularly in scenarios involving crucial data and assets. Consensus algorithms, by ensuring consistency among distributed nodes, provide a solid foundation for the trustworthiness and availability of blockchain.

Among these algorithms, Practical Byzantine Fault Tolerance (PBFT) [5] is a widely adopted consensus algorithm in blockchain applications in various domains, including finance, supply chain management, and governmental sectors [6–8]. For instance, in supply chain management, PBFT can be employed to verify and authorize each step from production to delivery, enhancing traceability and transparency in the supply chain.

Despite making substantial progress in addressing Byzantine fault tolerance, PBFT still faces several crucial challenges. Firstly, when the number of nodes is substantial, PBFT incurs higher communication overhead and latency, which can affect system performance and scalability, especially in high-frequency trading and large-scale systems. Secondly, the algorithm's fault tolerance concerning Byzantine nodes relies on assessing node stakes, and accurately evaluating node stakes remains a challenging issue. For example, in supply chain management, node stake assessment may be influenced by various factors like supplier history and delivery records.

Therefore, this research aims to address the aforementioned challenges by proposing an innovative approach that combines equity assessment and enhanced consistency hashing to improve the performance and security of PBFT. The contributions of this article are as follows.

1. Designing a more precise and comprehensive equity assessment mechanism that considers various factors such as historical behavior, contribution, and reputation. We will introduce a more complex equity calculation model to accurately reflect a node's value within the system.

2. Innovatively designing a distributed primary node selection mechanism to mitigate single-point failures and centralization risks. The introduction of a group hash ring will enable multiple nodes to participate in the primary node selection process, enhancing the algorithm's robustness and security.

3. The introduction of a delayed view change mechanism that dynamically adjusts the timing of view changes based on system load and node status. This helps to avoid unnecessary switching and reduces system overhead.

The remaining sections of this paper are organized as follows: In Section 2, we will delve into the preliminary work related to this research. Section 3 will provide a detailed overview of our proposed SC-PBFT design. In Section 4, we will demonstrate the effectiveness and performance advantages of our approach through experiments and analysis. Finally, in Section 5, we will conclude the entire paper.

2 Related work

In recent research of Byzantine fault tolerance, several pioneering papers have attracted considerable attention. Gao et al. [9] have introduced T-PBFT, a multi-stage consensus algorithm optimized through the utilization of the Eigen-Trust model. Their primary objective is to significantly enhance the performance and efficiency of Byzantine fault-tolerant consensus mechanisms. Li et al. [10] have proposed a scalable multi-layer PBFT consensus mechanism, with the aim of addressing performance bottlenecks in blockchain systems relying on proof of work (PoW). Moreover, Lao et al. [11] have formulated G-PBFT, a consensus protocol meticulously designed for IoT-blockchain applications. It takes into meticulous consideration the distinctive requirements of this domain and harnesses the inherent characteristics of fixed IoT devices to reduce exposure to malicious nodes, while concurrently improving consensus efficiency. Mišić et al. [12] have introduced a PBFT ordering service custom-tailored for permissioned blockchain environments. In stark contrast to current PBFT implementations, they have innovatively presented a single entry point for ordering services, thereby enabling each ordering node to function as an entry point and execute the consensus process for the inclusion of new records within the distributed ledger. Li et al. [13] have proposed the Extensible-PBFT (EPBFT) consensus algorithm, which can adapt its steps to achieve consensus based on the network environment of the system. By incorporating the use of Verifiable Random Functions (VRF) for consensus node election, EPBFT becomes suitable for dynamic networks. Furthermore, Xu et al. [14] have propounded SG-PBFT, a solution that achieves heightened consensus efficiency through the optimization of the PBFT consensus process and the application of a fractional grouping

mechanism. This approach alleviates the strain on central servers and diminishes the vulnerability to single-node attacks. Tang et al. [15] have introduced tPBFT, which incorporates a trust and equity scoring mechanism among nodes. This integration facilitates the dynamic selection of consensus nodes, consequently enhancing system security and bolstering trustworthiness.

Yang et al. [16] have introduced novel node decision broadcast and threshold voting count models, which notably enhance the fault tolerance of consensus nodes. Through the amalgamation of these models, the authors have conducted joint failure analyses of nodes, thus ensuring that the system's fault tolerance surpasses the critical threshold of one-third when compared to traditional PBFT algorithms. In juxtaposition to traditional PBFT algorithms, Xu et al. [17] have proposed ABC-GSPBFT, which introduces a grouping scoring mechanism and an artificial bee colony optimization consensus process. These sophisticated optimizations are expressly aimed at enhancing consensus efficiency, curtailing communication overhead, and fortifying the reliability of consensus nodes. Additionally, Xie et al. [18] have proffered a node election methodology grounded in the Probabilistic Linguistic Term Set (PLTS) for PBFT consensus. This method demonstrably enhances the efficiency of consensus attainment by introducing the concept of confidence intervals (PLTS-CI) to represent uncertain and intricate voting information during the node election process.

Lastly, Liu et al. [19] have introduced P-PBFT, a seminal contribution that addresses prevailing issues such as elevated latency, excessive system overhead, and constrained support for small-scale applications. The algorithm's optimization methodology involves the grouping of large-scale network nodes based on their response speeds, subsequently conducting group-based consensus. This innovative approach substantially diminishes inter-node communication overhead and augments consensus efficiency. Furthermore, Zhang et al. [20] have proposed an optimization scheme for blockchain consensus algorithms, grounded in genetic algorithms. This approach meticulously utilizes genetic algorithms to iteratively select consensus node groups characterized by exceptional performance indicators through the PBFT consensus process. These high-performance nodes are inherently regarded as more reliable and can be effectively harnessed in the formation of consensus node groups. Li et al. [21] have introduced the "Mandala" model, which is predicated on a "Mesh-and-Spoke Network" architecture, meticulously organizing nodes into discrete layers and regulating communication protocols among these layers. This innovative network structure markedly enhances network transmission efficiency and possesses the inherent potential to address the prevalent issue of low transmission efficiency in blockchain networks.

3 SC-PBFT Algorithm Framework

In this section, we present the design of the SC-PBFT (Stake Evaluation and Consistent Hashing Practical Byzantine Fault Tolerance) algorithm. Our algorithm aims to improve the PBFT consensus algorithm by incorporating stake evaluation and consistent hashing techniques. The following subsections outline the key components of the SC-PBFT algorithm.

3.1 Accurate Stake Evaluation Mechanism

When it comes to the PBFT algorithm, a group of supervisory nodes is required to oversee the entire consensus process. Typically, nodes that have staked a certain number of tokens in the network are qualified to become supervisory nodes. In this paper, we employ a mechanism based on node performance to select these supervisory nodes, ensuring that only nodes with a vested interest in network security can participate and oversee the consensus process.

Simultaneously, we introduce the concept of Shapley value [22], which is a method used to distribute rewards among cooperative members in game theory. Incorporating this concept into the blockchain consensus mechanism encourages nodes to actively participate in consensus and ensures a fair and equitable distribution of rewards [23, 24]. This approach helps determining the value and contribution of each participant in cooperative gameplay, thereby enhancing the efficiency and fairness of the entire consensus process.

In the context of blockchain consensus mechanisms, particularly when employing a stake-based

evaluation mechanism to select supervisory nodes, the Shapley value can be applied as follows:

1. Firstly, based on a credit assessment model, identify the top 10% of nodes with high credit scores from among all nodes. Assign unique identifiers to these nodes and form them into a credit node set N.

$$N = \{\text{node}_1, \text{node}_2, \ldots, \text{node}_n\} \tag{1}$$

2. During each consensus game, all nodes compete for the right to record transactions and validate new blocks. Identify the top-performing 10% of nodes that excel in promptly responding to and validating new blocks. Assign unique identifiers to these nodes and form them into a timely validation set M.

$$M = \{\text{node}_1, \text{node}_2, \ldots, \text{node}_m\} \tag{2}$$

3. Take the intersection of the credit node set N and the timely validation set M to create a new set W. This set comprises nodes that possess both excellent creditworthiness and exceptional performance in the validation process.

4. To determine the value of each alliance (subset Z), define a value function $v(Z)$. The function $v : 2^N \to \mathbb{R}$ is defined for every set (a subset of W, where W is the set of nodes with both good creditworthiness and outstanding validation performance), and it is specified that $v(\emptyset) = 0$.

5. Let W represent the alliance of creditworthy nodes. Therefore, for any credit alliance belonging to $W \subseteq 2^N$, its value function $v(Z)$ can be calculated based on the definition:

$$v(Z) = \begin{cases} \alpha \cdot \theta(Z) + \beta \cdot \psi(Z) + \gamma \cdot \omega(Z), & Z \in W \\ 0, & Z \notin W \end{cases} \tag{3}$$

6. Within the alliance node set, each node's Shapley value is calculated based on the formation of alliances and the value function. This calculation reflects the node's relative value in different alliances.

$$\phi_i(N, v) = \frac{1}{|N|!} \sum_{Z \subseteq N \setminus \{i\}} \frac{|Z|!}{(|N| - |Z| - 1)!} \times [v(Z \cup \{i\}) - v(Z)] \tag{4}$$

7. After introducing the time factor, a node's response time is considered as a factor influencing profit distribution. The node's Shapley value calculation is based on the time factor to reflect the node's performance and speed in the consensus process. The node's Shapley value calculation formula is as follows:

$$\phi_i^T(N, V) = \frac{\phi_i(N, V)}{T_e - T_s} \tag{5}$$

where: T_s represents the start time when a new block is prepared and broadcasted to the entire network, T_e represents the end time when online nodes complete the legality verification of the new block. $T_e - T_s$ represents the time elapsed from the broadcast of the new block to the completion of verification. Let $T = T_e - T_s$, where a smaller value of T indicates faster block verification by the node, indicating a higher level of proactiveness in node verification. Therefore, a smaller denominator in the fraction results in a larger Shapley value for the node, leading to higher rewards for the node's verification efforts.

8. Nodes with the right to validate transactions and create new blocks receive block rewards denoted as R. According to the nodes' creditworthiness, the block rewards are redistributed. Therefore, the ultimate benefit obtained by node i within the alliance is determined as follows:

$$R_i = R \cdot \phi_i^T(N, V) \cdot \frac{\text{credit}_i}{\sum_{j=1}^{\text{num}_W} \text{credit}_i} \tag{6}$$

3.2 Innovative Distributed Primary Node Selection

This section outlines the selection and grouping mechanism of the SC-PBFT consensus algorithm illustrated in Figure 1. For the meaning of the related symbols, please refer to Table 1.

To address the challenges of single-point failure and centralization risk, our proposed mechanism leverages the concept of group hashing rings, allowing multiple nodes to participate in the primary node selection process [25–29].

Assuming the blockchain system comprises a total of N nodes, among which N_c nodes are designated for consensus, while the remaining N_0 nodes are regular participants. Additionally, considering the system's fault tolerance, assuming f Byzantine nodes exist among these N_c consensus nodes.

Table 1 symbols used in SC-PBFT consensus algorithm.

Symbol	Meaning
N	Total Number of Nodes
f	Maximum Fault Tolerance of Nodes
N_0	Regular Nodes
N_c	Number of Consensus Nodes
N_s	Number of Supervisory Nodes
N_L	Leader Nodes
G_i	Group Number
k	Number of Nodes in a Consensus Group

To ensure system reliability, it is imperative to satisfy the condition $N_c \geq 3f + 1$. Here, we set $N_c = 3f+1$, representing the number of consensus nodes.

Subsequently, we will employ a hash consensus algorithm to group these N_c consensus nodes, with each group consisting of k nodes. This process will result in N_c/k groups. The following are the specific steps:

1. Initialization of the Hash Ring: Firstly, each node computes the hash value of its IP address, such as $hash(node_ip)$, and then maps these hash values onto a range from 0 to $2^{32} - 1$ within a hash ring. Notably, regulatory nodes form a distinct small hash ring, while all nodes collectively establish a large hash ring. The primary consensus process will occur within the confines of the large hash ring.

2. Selection of Starting Node and Leader Node: At the beginning of the initial consensus, we randomly select a supervisory node, denoted as N_{s_1}, from the small hash ring as the starting point of the consensus. Then, we choose $3f + 1$ nodes in a clockwise direction as consensus nodes, while the remaining nodes are marked as ordinary nodes. For these nodes, we record their respective trust values and identifiers.

3. Formation of Groups and Election of Primary Nodes: Within the large hash ring, nodes are categorized into different groups based on their hash mapping values. Each group calculates a specific hash value, such as $\text{hash}(\text{hash}(IP_{N_L}) + k + G_i)$, and selects the node closest to this hash value as the primary node for the group. This primary node assumes the leadership role in guiding the consensus process for the respective group. It is worth noting that the primary node of the first group concurrently holds the position of the leader node, with its IP address being N_{s_1}.

4. Consensus Process: Once the system is operational, and the initial consensus is concluded, the system initiates the re-selection of leader nodes and other relevant nodes, while excluding any potentially malfunctioning or malicious nodes. This process involves the computation of a new hash mapping point, such as $\text{hash}(\text{hash}(preIP_{N_L}) + k + preBlock_{hash})$. Subsequently, consensus nodes and group formations are re-established following a similar procedure. In each hash group, we select the node with the highest trust value as the primary node. In cases where multiple nodes possess identical trust values, the primary node is determined based on proximity to the hash mapping point with the highest trust value. It is important to note that $preIP_{N_L}$ denotes the IP address of the preceding leader node, while $preBlock_{\text{hash}}$ represents the hash value of the previous block.

The above process provides a detailed explanation of how to group consensus nodes and select master nodes within a blockchain system. These series of steps contribute to ensuring the high quality and reliability of the blockchain system while meeting the requirements for fault tolerance.

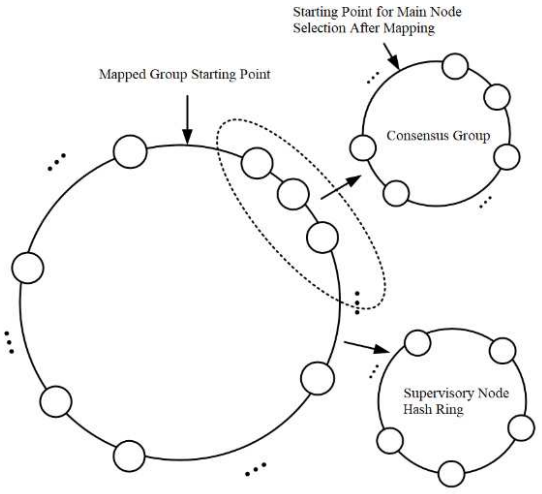

Fig. 1 The uniform hash algorithm is used for grouping.

3.3 Design of the SC-PBFT Algorithm

SC-PBFT is an improvement over the traditional PBFT algorithm, as shown in Figure 2, aiming to enhance both performance and security. Here is an overview of the SC-PBFT algorithm process:

1. Request Phase: The client initiates the process by sending a message to the primary node, signing it as sign(REQUEST,operation,timestamp,clientID). The REQUEST message includes specific details such as the operation to be performed, appended timestamp, and the client's identifier.

2. Pre-Prepare Phase: The primary node sends a message to the other replica nodes, signed as sign(<PRE-PREPARE,view,sequence,digest>,message). Here, view represents the view number, sequence denotes the sequence number of the pre-prepare message, which should fall within a specific range (e.g., [h, H]), and digest serves as the message digest, typically used for verifying message integrity. The message contains the actual content of the message to be processed.

3. Prepare Phase (Node Verification): When replica $node_i$ receives the PRE-PREPARE message from the master node, the node splits the incoming message. This is done with the purpose of validating the signature of the master node and the signature provided by the client, ensuring the transactions within the message are correctly ordered based on their timestamps and guaranteeing that no PRE-PREPARE messages sharing the same sequence number but bearing different signatures within the same view are received. If the master node's integrity is confirmed, the node proceeds to transmit its own verified transaction information's hash value, along with its own signature, back to the master node. However, in cases where suspicions arise regarding the trustworthiness of the master node, the node may consider requesting a rotation of the master node.

4. Commit Phase: If a node successfully validates during the pre-prepare phase, it can transition into the prepare state to continue message validation and commit operations. When a node receives PREPARE messages from $2f + 1$ nodes that have been successfully validated, it proceeds to send COMMIT messages to other nodes. These COMMIT messages are of the form: sign(COMMIT,view,sequence,digest,i), and the node keeps a record of these messages for future reference.

5. Reply Phase: Once a node receives COMMIT messages from $2f + 1$ nodes that have been successfully validated, it can proceed to execute the client's request operation and generate a response message: <REPLY,view,timestamp,clientID,i,result>. Here, result represents the outcome of the operation's execution.

In the SC-PBFT consensus algorithm, nodes receive corresponding rewards upon completing consensus, while nodes that fail to complete the consensus will be penalized accordingly. These scores are used to measure the nodes' reputation, and they undergo moderate decay over time and as reward scores change.

Additionally, the system regularly evaluates the performance of nodes, typically after every 20 rounds of consensus, to identify the group of nodes with the poorest performance. These nodes will be replaced to maintain the high quality and reliability of the consensus network. Lastly, when a node's reputation score falls below 50 points, they will lose their eligibility to become consensus nodes.

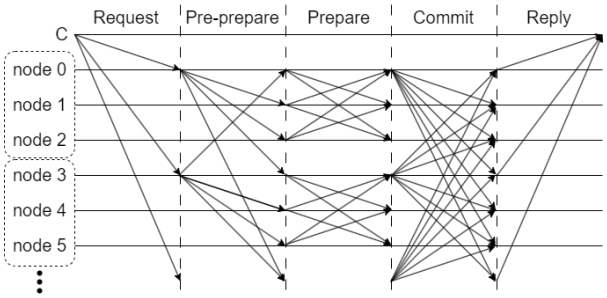

Fig. 2 The SC-PBFT Consensus Protocol.

3.4 Optimizing View Protocol

PBFT requires maintaining a shared state where all nodes take consistent actions. To achieve this, three fundamental protocols are run, including the consensus protocol, checkpoint protocol, and view change protocol. View change is a crucial component of the consensus protocol. In Byzantine fault-tolerant consensus algorithms, view change

is used to handle scenarios involving the failure of a primary node or Byzantine faults to ensure the continuity of consensus. View change is an operation with high overhead as it involves communication and state synchronization among nodes [30–33]. This paper reduces communication overhead by delaying the timing of view change and batch processing of requests. The process is illustrated in Figure 3.

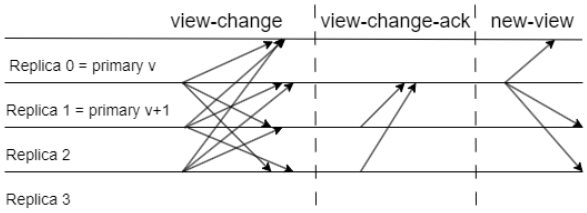

Fig. 3 View Change Process.

1. View-Change Phase: After identifying the need for a view change, instead of immediately executing the switching operation, the system enters a waiting phase that allows ongoing processing of requests in the current view. A delay timer is set to postpone the triggering of the view-change. This timer is dynamically adjusted based on factors such as system load, network conditions, and node status. Prior to triggering the view change, the collection of pending requests continues. These requests will be processed together when the switching is triggered, thereby reducing the number of communication rounds and the overhead of state synchronization.

2. View-Change-ACK Phase: Replica nodes create View-Change messages <VIEW-CHANGE,v,blockHeight,hash(TXs),TXs>. Once these View-Change messages are broadcasted to the network, backup nodes can asynchronously process and validate their effectiveness.

Firstly, backup nodes verify the correctness of the signature of the VIEW-CHANGE message. Secondly, they confirm that v is greater than the current number of the replica node by 1. This validation ensures that VIEW-CHANGE messages are submitted in the correct order. Finally, if hash(TXs) is not empty, backup nodes verify whether it matches the hash value of the TXs collection. This helps ensuring that the submitted transactions are complete and have not been tampered with.

While waiting for VIEW-CHANGE messages, backup nodes can continue processing client requests without having to wait for the view-change trigger for an extended period.

3. New-View Phase: When there are $2f + 1$ VIEW-CHANGE messages in the view-change set, the network creates message <NEW-VIEW,v+1, view_set>. In this message, NEW-VIEW serves as an indicator for transitioning to a new view, v+1 represents the number of the new view, and view_set contains configuration details for the new view.

Within the NEW-VIEW message, the primary node may include pending requests that were collected during the View-Change phase. By batch-processing these requests, the goal is to reduce communication overhead and minimize the frequency of state synchronization. Upon receiving the NEW-VIEW message, backup nodes asynchronously handle these requests without waiting for responses from other nodes. This optimization reduces the transition time, ensuring that the new primary node can promptly begin processing requests. The delayed view change algorithm is shown in Algorithm 1.

Algorithm 1 Delayed ViewChange Algorithm.

Require: $n \geq 0 \vee x \neq 0$ Ensure $y = x^n$

while $Message_{view-change}$ **do**
  if count($Message_{view-change}$) $\geq 2f + 1$ **then** // change the current view number
    $view_{cur}$ += 1 //broadcast the new view message
    $Message_{new} \Leftarrow view_{cur}$
    $N_0 \Leftarrow Message_{new}$
    $fuc \Leftarrow request_{client}$
    if $N_0.Message_{view-change}$ **then** //verify the signature of primary node and client
      Sig(N_c) and Sig(client) $\Leftarrow fuc$
      $Message_{view-change}.timestamp \Leftarrow fuc$
      Sig($Message_{Pre-Prepare}$) $\Leftarrow fuc$
    end if
  end if Update(timer)
end while

4 Experiment and Analysis

The simulation experiments are based on the Java programming language and implemented on a blockchain system running on Windows 10, equipped with an Intel i7-11375H CPU and 16 GB of memory. The proposed SC-PBFT algorithm is validated in this system, and an evaluation is conducted on various aspects, including security, consensus latency, communication overhead, and throughput, for different numbers of nodes. The experimental configuration details are presented in Table 2.

Table 2 Configuration of Experimental System.

Object	Configuration
CPU	11th Gen Intel(R) Core(TM) i7-11375H @ 3.30GHz
Operating System	Ubuntu 20.04 LTS
Memory	16GB
Software Environment	MATLAB, GO

4.1 Security Analysis

4.1.1 Node Security

The SC-PBFT algorithm ensures the authenticity of messages and the trustworthiness of senders through a digital signature mechanism. Each node digitally signs messages when sending them, and other nodes can verify the signatures to determine the legitimacy of the messages. This prevents malicious nodes from forging messages, ensuring the security of nodes.

Furthermore, the SC-PBFT algorithm uses a random rotation mechanism for the selection of consensus nodes, avoiding the centralization issue in node selection and reducing the risk of collusion among nodes. Node participation and exit also undergo certain security verification to ensure the trustworthiness of node identities and behaviors.

4.1.2 Consensus Security

During the consensus process, the SC-PBFT algorithm employs a three-phase message propagation mechanism, namely the pre-prepare, prepare, and commit phases. In each phase, nodes need to receive a certain number of messages before proceeding to the next step, ensuring the correctness and consistency of messages. This mechanism prevents malicious behavior by Byzantine nodes from interfering with the consensus process.

In the SC-PBFT algorithm, each node requires receiving confirmation messages from more than 2/3 of the nodes to consider consensus achieved. This requirement ensures that the system can maintain the security of consensus even in the presence of a small number of Byzantine nodes. Regarding the number of Byzantine nodes, the SC-PBFT algorithm can tolerate up to 1/3 of the nodes engaging in malicious behavior, thereby guaranteeing the correctness of consensus.

Overall, the SC-PBFT algorithm demonstrates excellent performance in both node security and consensus security. Through digital signatures and message propagation mechanisms, node behavior and messages are safeguarded, preventing forgery and malicious actions. Through a multi-phase consensus process, the system can withstand malicious interference by Byzantine nodes, ensuring the correctness and security of consensus.

4.2 Communication Overhead Analysis

In the PBFT consensus process, there are three main stages: pre-prepare, prepare, and commit. For the pre-prepare stage, the primary node broadcasts messages to all communication counts of $N-1$ (where N is the number of nodes). In the prepare stage, nodes send validation messages to all nodes except themselves, resulting in a communication count of $(N-1)^2$. In the commit stage, each node broadcasts commit messages to other nodes, resulting in a communication count of $N*(N-1)$. Therefore, the total communication complexity T_1 for PBFT is calculated as follows:

$$\begin{aligned} T_1 &= N-1+(N-1)^2+N\cdot(N-1) \\ &= 2N\cdot(N-1) \end{aligned} \tag{7}$$

The SC-PBFT algorithm introduces a grouping phase and calculates node credit values during the consensus process. In this algorithm, clients initiate proposals to management nodes with a communication count of C (where C is the number of nodes in the consensus set). During the prepare stage within the consensus set, management nodes broadcast messages to members of the consensus set, resulting in a communication

count of $[(N-1)/(C-1)] * C$. In the commit stage within the consensus set, each member of the consensus set sends messages to the management node for submission, resulting in a communication count of $[(N-1)/(C-1)] * C$. During the non-consensus set prepare stage, management nodes broadcast messages to other management nodes except themselves, with a communication count of $(C-1)*C$. In the non-consensus set commit stage, each management node verifies received messages and broadcasts them to other management nodes, with a communication count of $(C-1) * C$. The total communication complexity T_2 for SC-PBFT is calculated as follows:

$$\begin{aligned} T_2 &= C + 2\left(\frac{N-1}{C-1}\right)C + 2(C-1)C \\ &= (2C-3)C + 2(N-1) \end{aligned} \tag{8}$$

T_1 represents the communication complexity when all nodes participate in consensus. T_2 represents the communication complexity when nodes in the consensus set participate in consensus. Comparing the communication complexities T1 and T2, we can draw the following conclusions: Since $C < N$, the communication complexity T_2 is generally much smaller than T_1. To provide a comparative analysis of recent improvements to the PBFT algorithm, we consider GPBFT and SG-PBFT for comparison. The communication complexity for GPBFT (T_3) is represented as

$$T_3 = N^2 + N - 1 \tag{9}$$

and for SG-PBFT (T_4), it is represented as

$$T_4 = \frac{N^2}{4} - 1 \tag{10}$$

We conducted experiments and simulations for these four consensus processes, and in the experiments, we idealized the SC-PBFT for fixed-point simulation. The number of nodes was selected as 16, 36, 64, 100, 144, 196, 400, 1024, 1600, 2500, 4096, 4900, 6400, 8100, and 10000.

The results in Figure 4 depict the communication complexity of PBFT when all nodes participate in consensus. As the number of nodes N increases, the communication complexity exhibits quadratic growth. In large-scale networks, this leads to a sharp increase in communication complexity and a subsequent performance decrease. GPBFT, a consensus algorithm similar to PBFT, also exhibits communication complexity that grows quadratically with N. In large-scale networks, its communication complexity is similar to PBFT, showing an exponential growth trend.

In summary, in large-scale networks, SC-PBFT and SG-PBFT may be better choices to reduce communication overhead and improve performance. However, SG-PBFT may sacrifice a certain level of security. The proposed SC-PBFT in this paper significantly reduces communication complexity by choosing an appropriate number of nodes in the consensus set, demonstrating superior performance in large-scale networks.

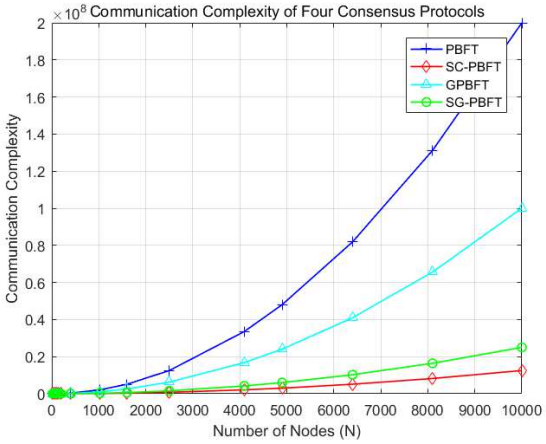

Fig. 4 Communication overhead comparison.

4.3 Latency Analysis

Consensus latency refers to the time required from when a client initiates a transaction request to when the client receives confirmation of the completed transaction. Low latency indicates that the consensus algorithm executes with less time, reducing the likelihood of blockchain forks, and making the system more secure and efficient. The latency formula is expressed as follows:

$$Delay = T_c - T_r \tag{11}$$

where T_c represents the transaction confirmation time, and T_r represents the transaction generation time. We conducted each experiment group 100 times and averaged the results. The number of Byzantine nodes in the system follows the condition $f(n-1)/3$, with some exceptions excluded.

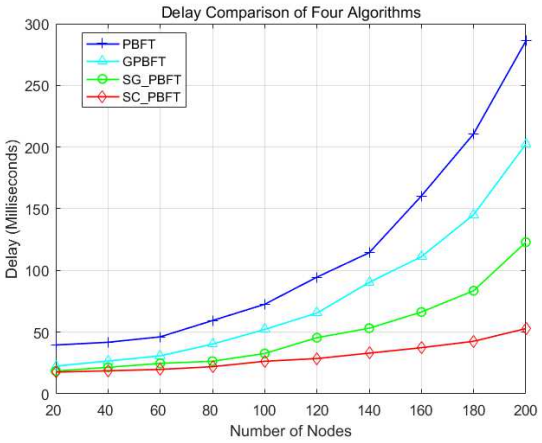

Fig. 5 Transaction delay comparison.

As can be observed from Figure 5, with an increase in the number of nodes, the consensus latency for all four algorithms gradually increases. PBFT has the highest consensus latency among different node counts. SC-PBFT algorithm performs better than PBFT overall because SC-PBFT selects nodes with higher reputation as primary nodes based on a stake evaluation mechanism. The identity of primary nodes is less susceptible to malicious prediction attacks, resulting in a more trusted consensus node cluster. This reduces the frequency of view changes, resulting in lower consensus latency and higher consensus efficiency.

When there are fewer nodes, SC-PBFT algorithm introduces a consistency hash grouping strategy, causing slightly higher consensus latency compared to the SG-PBFT algorithm. However, when the number of nodes exceeds 60, malicious nodes are gradually eliminated from the SC-PBFT algorithm's consensus cluster, reducing the number of consensus nodes. Consequently, the consensus latency becomes lower than that of the SG-PBFT algorithm.

When there are a large number of nodes, SG-PBFT algorithm's latency is slightly higher than SC-PBFT algorithm. This is because SG-PBFT simplifies the consensus phase when there are no Byzantine nodes, resulting in slightly higher latency compared to SC-PBFT algorithm.

4.4 Throughput Analysis

Throughput refers to the number of transactions or requests a system can process in a unit of time. In consensus algorithms, throughput is one of the key metrics for measuring system performance. For blockchain or distributed systems, throughput gauges the system's processing capacity, specifically the number of transactions or operations it can complete per second.

$$TPS = \frac{Transactions_{\Delta t}}{\Delta t} \tag{12}$$

The term $Transactions_{\Delta t}$ represents the number of transactions completed within a time interval Δt, where Δt represents the time interval between transaction submission and publication on the blockchain.

In this section of the experiment, the throughput of the four algorithms is evaluated by having clients to send 1000 transaction requests. The average of multiple experimental data points is taken to ensure the validity of the experimental results.

As shown in Figure 6, the experimental results indicate that the SC-PBFT algorithm performs as the optimal algorithm in the provided data. It exhibits high throughput across various numbers of nodes and is suitable for large-scale networks. Furthermore, its performance remains relatively stable as the number of nodes increases. In contrast, PBFT is a Byzantine fault-tolerant algorithm that maintains security and consistency in scenarios with up to f node failures. It performs well in small-scale node networks but experiences a gradual decrease in throughput as the number of nodes increases, making it less suitable for large-scale networks. GPBFT is an extension and improvement of PBFT, offering better fault tolerance to handle more failure scenarios. However, it is also not well-suited for large-scale networks. SG-PBFT, a simplified version of GPBFT, is

designed to streamline the algorithm implementation and enhance performance. SG-PBFT demonstrates higher throughput in small-scale networks but may exhibit poorer performance in large-scale networks. SC-PBFT, on the other hand, increases the algorithm's throughput and performance by selecting core nodes. Across various numbers of nodes, SC-PBFT consistently displays the highest throughput and performance. It operates effectively in large-scale networks and maintains stable performance as the number of nodes increases.

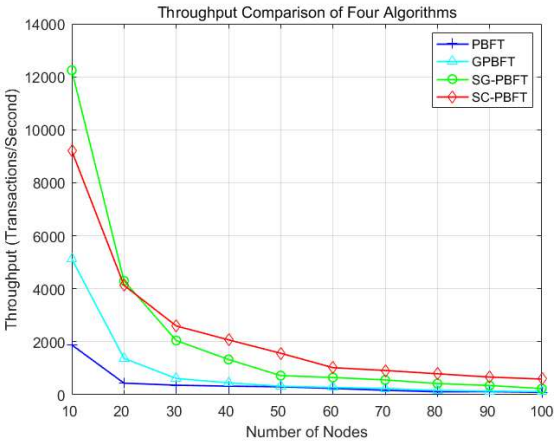

Fig. 6 Throughput comparison experiment.

5 Conclusion

In this study, we have explored how to improve Byzantine fault-tolerant protocols by introducing a stake evaluation mechanism and a consistency hashing algorithm. This improvement aims to enhance the security and performance of blockchain systems in a Byzantine attack environment, thereby increasing trust in honest nodes, reducing the impact of malicious behavior, and optimizing network scalability.

We analyzed the benefits of introducing the Shapley value for stake evaluation, which encourages honest behavior through a reputation system and increases the difficulty of Byzantine attacks. Additionally, we conducted research on improving the consistency hashing algorithm to enhance network scalability and reduce node communication overhead. These improvement measures have a positive impact on enhancing the overall performance and security of the network. By encouraging stakeholders to participate in the consensus process and establish reputations, we reduce the likelihood of successful Byzantine attacks. Furthermore, the enhancements to the consistency hashing algorithm improve the network's scalability, enabling it to better handle the growing number of nodes and data loads. These efforts contribute to maintaining system stability while reducing potential risks, making the Byzantine fault-tolerant protocol more robust in the face of various challenges.

Through experimental validation, we observed that the improved protocol exhibits greater stability when facing Byzantine attacks, and the influence of malicious nodes is mitigated to some extent. Moreover, under normal circumstances, network performance is enhanced, and latency and communication overhead are relatively reduced. These results demonstrate the potential and practicality of the proposed improvement in the field of Byzantine fault tolerance.

However, it's important to note that the improved methods may have limitations in certain extreme scenarios. In specific attack scenarios, malicious nodes may still bypass the defense of the reputation system through coordinated behavior, or the consistency hashing algorithm may perform poorly in certain network topologies. Therefore, we need to consider various factors and strike a balance between security and performance in practical applications, adjusting and optimizing based on specific circumstances.

In conclusion, the Byzantine fault-tolerant improvement approach based on stake evaluation and consistency hashing is a promising research direction that provides effective means to enhance the security and performance of blockchain systems. In future work, we can further explore more complex attack models, more efficient reputation mechanisms, and adaptability adjustments for different network topologies to achieve a more robust and efficient Byzantine fault-tolerant protocol.

Acknowledgements The authors would like to sincerely thank the editor and the anonymous reviewers for their valuable suggestions to improve the quality of this work.

Author contribution All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Guangfu Wu, Xin Lai and Daojing He. The first draft of the manuscript was written by Xin Lai Xiaoyan Fu and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This work was supported by the the State Key Laboratory of Cryptology (Grant No. MMKFKT202123), by the Key Program of Natural Sciences Foundation of Jiangxi Province, China (Grant No. 20212ACB202003), by the National Natural Science Foundation of China (Grant No.11461031).

Data availability Not applicable.

Declarations

Ethics approval The submitted works are original and have not been published elsewhere in any form or language (partially or in full), nor are they under consideration by another publisher. This manuscript has no plagiarism, fabrication, falsification, or inappropriate manipulation.

Consent to publish All the authors agree to publication in Peer-to-Peer Networking and Applications.

Competing interests The authors declare no competing interests.

References

[1] Zheng, Z., Xie, S., Dai, H.-N., Chen, X., Wang, H.: Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services **14**(4), 352–375 (2018) https://doi.org/10.1504/IJWGS.2018.095647

[2] Tapscott, A., Tapscott, D.: How Blockchain Is Changing Finance (2017)

[3] Dutta, P., Choi, T.-M., Somani, S., Butala, R.: Blockchain technology in supply chain operations: Applications, challenges and research opportunities. Transportation Research Part E: Logistics and Transportation Review **142**, 102067 (2020) https://doi.org/10.1016/j.tre.2020.102067

[4] Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: MedRec: Using Blockchain for Medical Data Access and Permission Management. In: 2016 2nd International Conference on Open and Big Data (OBD), pp. 25–30 (2016). https://doi.org/10.1109/OBD.2016.11

[5] Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems **20**(4), 398–461 (2002) https://doi.org/10.1145/571637.571640

[6] Qu, J.: Blockchain in medical informatics. Journal of Industrial Information Integration **25**, 100258 (2022) https://doi.org/10.1016/j.jii.2021.100258

[7] Xu, X., Zhu, D., Yang, X., Wang, S., Qi, L., Dou, W.: Concurrent Practical Byzantine Fault Tolerance for Integration of Blockchain and Supply Chain. ACM Transactions on Internet Technology **21**(1), 7–1717 (2021) https://doi.org/10.1145/3395331

[8] Xu, J., Hua, C., Zhang, Y.: A Blockchain-Based Framework for Supervision of Livelihood Issues: Proof of Concept With Optimized Consensus. IEEE Access **11**, 73414–73434 (2023) https://doi.org/10.1109/ACCESS.2023.3295696

[9] Gao, S., Yu, T., Zhu, J., Cai, W.: T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm. China Communications **16**(12), 111–123 (2019) https://doi.org/10.23919/JCC.2019.12.008

[10] Li, W., Feng, C., Zhang, L., Xu, H., Cao, B., Imran, M.A.: A Scalable Multi-Layer PBFT Consensus for Blockchain. IEEE Transactions on Parallel and Distributed Systems **32**(5), 1146–1160 (2021) https://doi.org/10.1109/TPDS.2020.3042392

[11] Lao, L., Dai, X., Xiao, B., Guo, S.: G-PBFT: A Location-based and Scalable Consensus

Protocol for IoT-Blockchain Applications. In: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 664–673 (2020). https://doi.org/10.1109/IPDPS47924.2020.00074

[12] Mišić, J., Mišić, V.B., Chang, X., Qushtom, H.: Adapting PBFT for Use With Blockchain-Enabled IoT Systems. IEEE Transactions on Vehicular Technology **70**(1), 33–48 (2021) https://doi.org/10.1109/TVT.2020.3048291

[13] Li, Y., Wang, Z., Fan, J., Zheng, Y., Luo, Y., Deng, C., Ding, J.: An extensible consensus algorithm based on pbft, 17–23 (2019) https://doi.org/10.1109/CyberC.2019.00013

[14] Xu, G., Bai, H., Xing, J., Luo, T., Xiong, N.N., Cheng, X., Liu, S., Zheng, X.: SG-PBFT: A secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent Internet of vehicles. Journal of Parallel and Distributed Computing **164**, 1–11 (2022) https://doi.org/10.1016/j.jpdc.2022.01.029

[15] Tang, S., Wang, Z., Jiang, J., Ge, S., Tan, G.: Improved PBFT algorithm for high-frequency trading scenarios of alliance blockchain. Scientific Reports **12**(1), 4426 (2022) https://doi.org/10.1038/s41598-022-08587-1

[16] Yang, J., Jia, Z., Su, R., Wu, X., Qin, J.: Improved Fault-Tolerant Consensus Based on the PBFT Algorithm. IEEE Access **10**, 30274–30283 (2022) https://doi.org/10.1109/ACCESS.2022.3153701

[17] Xu, J., Zhao, Y., Chen, H., Deng, W.: ABC-GSPBFT: PBFT with grouping score mechanism and optimized consensus process for flight operation data-sharing. Information Sciences **624**, 110–127 (2023) https://doi.org/10.1016/j.ins.2022.12.068

[18] Xie, M., Liu, J., Chen, S., Xu, G., Lin, M.: Primary node election based on probabilistic linguistic term set with confidence interval in the PBFT consensus mechanism for blockchain. Complex & Intelligent Systems **9**(2), 1507–1524 (2023) https://doi.org/10.1007/s40747-022-00857-9

[19] Liu, S., Zhang, R., Liu, C., Shi, D.: P-PBFT: An improved blockchain algorithm to support large-scale pharmaceutical traceability. Computers in Biology and Medicine **154**, 106590 (2023) https://doi.org/10.1016/j.compbiomed.2023.106590

[20] Zhang, J., Yang, Y., Zhao, D., Wang, Y.: A node selection algorithm with a genetic method based on PBFT in consortium blockchains. Complex & Intelligent Systems **9**(3), 3085–3105 (2023) https://doi.org/10.1007/s40747-022-00907-2

[21] Li, J., Li, X., Zhao, H., Yu, B., Zhou, T., Cheng, H., Sheng, N.: MANDALA: A scalable blockchain model with mesh-and-spoke network and H-PBFT consensus algorithm. Peer-to-Peer Networking and Applications **16**(1), 226–244 (2023) https://doi.org/10.1007/s12083-022-01373-w

[22] Littlechild, S.C., Owen, G.: A simple expression for the shapley value in a special case. Management Science **20**(3), 370–372 (1973)

[23] Shen, M., Duan, J., Zhu, L., Zhang, J., Du, X., Guizani, M.: Blockchain-Based Incentives for Secure and Collaborative Data Sharing in Multiple Clouds. IEEE Journal on Selected Areas in Communications **38**(6), 1229–1241 (2020) https://doi.org/10.1109/JSAC.2020.2986619

[24] Du, Y., Wang, Z., Li, J., Shi, L., Jayakody, D.N.K., Chen, Q., Chen, W., Han, Z.: Blockchain-aided edge computing market: Smart contract and consensus mechanisms. IEEE Transactions on Mobile Computing (2022) https://doi.org/10.1109/TMC.2021.3140080

[25] Yin, H., Zhang, Z., He, J., Ma, L., Zhu, L., Li, M., Khoussainov, B.: Proof of continuous work for reliable data storage over permissionless blockchain. IEEE Internet of Things Journal **9**(10), 7866–7875 (2021) https://doi.org/10.1109/JIOT.2021.3115568

[26] Zhang, Y., Gan, Y., Li, C., Deng, C., Luo, Y.: Primary node selection based on node reputation evaluation for PBFT in UAV-assisted MEC environment. Wireless Networks (2023) https://doi.org/10.1007/s11276-023-03407-4

[27] Xie, M., Liu, J., Chen, S., Xu, G., Lin, M.: Primary node election based on probabilistic linguistic term set with confidence interval in the PBFT consensus mechanism for blockchain. Complex & Intelligent Systems **9**(2), 1507–1524 (2023) https://doi.org/10.1007/s40747-022-00857-9

[28] Tangsen, H., Li, X., Ying, X.: A Blockchain-Based Node Selection Algorithm in Cognitive Wireless Networks. IEEE Access **8**, 207156–207166 (2020) https://doi.org/10.1109/ACCESS.2020.3038321

[29] Wu, G., Li, E., Wei, T.: Multimaster node byzantine fault-tolerant consensus algorithm based on consistent hash algorithm. Computer **56**(11), 48–63 (2023) https://doi.org/10.1109/MC.2023.3255305

[30] Dinh, T.T.A., Liu, R., Zhang, M., Chen, G., Ooi, B.C., Wang, J.: Untangling Blockchain: A Data Processing View of Blockchain Systems. IEEE Transactions on Knowledge and Data Engineering **30**(7), 1366–1385 (2018) https://doi.org/10.1109/TKDE.2017.2781227

[31] Qiu, C., Yu, F.R., Yao, H., Jiang, C., Xu, F., Zhao, C.: Blockchain-Based Software-Defined Industrial Internet of Things: A Dueling Deep Q -Learning Approach. IEEE Internet of Things Journal **6**(3), 4627–4639 (2019) https://doi.org/10.1109/JIOT.2018.2871394

[32] Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: HotStuff: BFT Consensus in the Lens of Blockchain. arXiv (2019). https://doi.org/10.48550/arXiv.1803.05069

[33] Wu, G., Li, E., Wei, T.: Multimaster node byzantine fault-tolerant consensus algorithm based on consistent hash algorithm. Computer **56**(11), 48–63 (2023) https://doi.org/10.1109/MC.2023.3255305

Guangfu Wu was born in Yushan, Jiangxi, China, in 1977. He received the B.S. degree in mathematics education from Wuyi University, Guangdong, in 2000, the M.S. degree from the School of Mathematical Sciences, Xiamen University, Xiamen, in 2008, and the Ph.D. degree from the School of Information Science and Engineering, Xiamen University, in 2012. Since 2016, he has been an Assistant Professor with the School of Information Engineering, Jiangxi University of Science and Technology. He is the author of more than ten articles, and more than ten inventions. His research interests include coding theory and cryptography, blockchain, and artifificial intelligence. Dr. Wu became a member (M) of the Chinese Association for Cryptologic Research (CACR), in 2014.

Xin Lai is currently pursuing a Master's degree at the School of Information Engineering, Jiangxi University of Science and Technology, Jiangxi, China, with research interests in medical blockchain, consensus algorithms, and smart contracts.

Daojing He received the B.Eng. and M.Eng. degrees in computer science from the Harbin Institute of Technology, China, in 2007 and 2009, respectively, and the Ph.D. degree in computer science from Zhejiang University, China, in 2012. He is currently a Professor with the computer science department, Harbin Institute of Technology, Shenzhen, and also a Guest Professor with the Department of Information Engineering, Jiangxi University of Science and Technology. His research interest includes network and systems security. He is an Associate Editor or on the Editorial Board of some

international journals, such as IEEE Communications Magazine and the IEEE/KICS JOURNAL OF COMMUNICATIONS AND NETWORKS.

Sammy Chan received the B.E. and M.Eng. Sc. degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and the Ph.D. degree in communication engineering from the Royal Melbourne Institute of Technology, Australia, in 1995. From 1989 to 1994, he was with the Telecom Australia Research Laboratories, fifirst as a Research Engineer, and from 1992 to 1994 as a Senior Research Engineer and a Project Leader. Since December 1994, he has been with the Department of Electronic Engineering, City University of Hong Kong, where he is currently an Associate Professor.

Xiaoyan Fu is currently a master's student at the School of Information Engineering, Jiangxi University of Science and Technology, where her main research interests are blockchain technology, cryptography and information security.