Journal of Network Intelligence ©2024 ISSN 2414-8105
Taiwan Ubiquitous Information Volume **9**, Number **3**, August **2024**

Improved PBFT Consensus Algorithm Based on Threshold Signature and RabbitMQ

Xiang Xu, Chun-Yuan Liu*
School of Computer and Information Engineering
Heilongjiang University of Science and Technology, Heilongjiang Harbin 150022, China
xuxiang6195@163.com, 154539768@qq.com

*Corresponding author: Chun-Yuan Liu
Received August 15, 2023, revised November 22, 2023, accepted February 7 2024.

Abstract. *Due to its decentralized and tamper-proof features, blockchain is frequently employed in the financial, traceability, and distributed storage industries. The agreement algorithm, which is a crucial component of the blockchain distributed storage system, is responsible for maintaining data integrity and ensuring system security. It also has a significant impact on the blockchain system's transaction throughput and confirmation time. Compared with public chain, alliance chain has fewer nodes and access mechanism, so PBFT consensus algorithm based on voting is generally used. However, there are still many shortcomings in the practical application of this algorithm. To solve the problem, an improved Byzantine consensus algorithm (TRPBFT) based on threshold signature and RabbitMQ is proposed in this paper. The experimental findings demonstrate that the enhanced consensus algorithm significantly increases both consensus efficiency and data throughput.*
Keywords:blockchain; practical byzantine fault tolerance(PBFT); threshold signatures; rabbitmq; distributed storage

1. **Introduction.** The coal industry Internet system architecture usually realizes data sharing by building a big data platform to alleviate the phenomenon of "data island". This type of data storage typically relies on centralized data storage, which is susceptible to malicious network assaults or failures brought on by downtime and may result in security issues including central node failure, data loss, or manipulation [1,2]. Only the shortcomings in the data storage and sharing procedure in coal mines can be compensated for by the use of the consensus method, P2P technology, and encryption technology [3,4,5]. All these make the blockchain gradually become the research hotspot of coal mine data storage and sharing solutions in the coal industry Internet.

The attributes of blockchain include immutability, anonymity, persistence, and decentralization [6]. Being a distributed shared ledger system that links data blocks chronologically in a chain [7], blockchain integrates varieties of computer technologies such as distributed data storage, cryptography technology, smart contracts and consensus mechanism [8]. The previous block's hash value is included in each block of the blockchain, forming a linked list of data structures that not only assures that data cannot be modified simultaneously but also maintains the data's traceability [9]. In order to ensure that the data may be treated as being unchangeable once written, any changes to the block's data after it has been written require the consent of over 50 percent of the nodes in the system. This takes significant computer power and difficulties. Each node has a backup of the complete data in the system. If one node is attacked and the data is lost, the complete data can be retrieved from other nodes, which provides rapid recovery of the system data. The blockchain's autonomy is significantly increased by the use of smart contracts, which also reduces the need for human intervention and increases the usage of code and computers for the purpose [10]. Agreement mechanisms, one of the fundamental components of blockchain technology, are what allow dispersed systems to come to a single, agreed-upon agreement. Blockchain may be classified into three categories: public chain, private chain, the alliance chain, depending on the various application situations [11]. The public chain environment has a variable number of nodes and completely open to the outside world, and each node competes for the right to package data through consensus mechanisms such as mining. The private chain is not open to the public and the default nodes are honest, so the private chain consensus algorithm often has high consensus efficiency but low security, and can not resist the Byzantine general problem [12]. While the node count and status of the alliance chain are somewhat within your control, the algorithm security is stronger than that of the private chain. Compared with the ideal environment of private chain, alliance chain system is more suitable for information storage and sharing within or between enterprises in reality. Therefore, how to ensure the security of alliance chain consensus algorithm and make it have similar consensus efficiency and calculation and communication cost as private chain consensus algorithm, this problem has been the research focus of many scholars. Numerous better algorithms have been put forth one after the other in recent years, however there is no systematic improvement plan and the majority of the algorithms are intended to boost a certain algorithm's performance. Having studied blockchain technology in depth, the application scenarios of distributed systems are becoming more and more diverse, and the research on safe and efficient consensus algorithms suitable for different application scenarios also shows its necessity.

There are many similarities between alliance chain environment and traditional distributed system. Many consistency algorithms are applied to alliance chain as consensus algorithm. In 1990, Lamport [13] published a paper on "The Part-Time Parliament" and proposed the Paxos consensus algorithm, which is known as the most effective consistency algorithm. However, due to the lack of specific implementation details in the Paxos algorithm paper and the difficulty of understanding this algorithm, Not widely used. The Practical Byzantine Fault Tolerance (PBFT) algorithm was introduced in 1999 by Castro and Liskov [14]. By reducing the temporal complexity of the procedure to O(n2), this algorithm enhances the initial Byzantine Fault Tolerance (BFT) algorithm and can be used in real-world situations. This algorithm can ensure that the system is secure when there are no more than 1/3 Byzantine nodes in the cluster, but Byzantine nodes may still act as the primary nodes and affect the consensus process of the system. In 2014, Ongaro and Ousterhout [15] proposed the Replicated and fault tolerant (Raft) algorithm. Raft consensus algorithm was proposed to provide better understanding of the consistent algorithm. In the process, it resolves the issue that the Paxos algorithm is overly complex, and the author also demonstrates that the method is secure and equally as effective as Paxos. Leaders candidature as well as logs duplication are the two fundamental stages of the Raft consensus method. The Raft consensus algorithm's procedure is very clear and concise with high understandable, so it is easy to implement in practical application scenarios. However, the RAFT consensus algorithm cannot be used in a setting with Byzantine nodes, and consensus cannot be reliably achieved when there are malevolent nodes present. The PBFT consensus algorithm can be used in a setting with Byzantine nodes, but it has some drawbacks, including poor consensus efficiency, poor scalability, and unreliable master node selection. Therefore, this work offers a more efficient useful Byzantine consensus algorithm TRPBFT based on RabbitMQ and threshold signature in order to solve the issues with the current PBFT consensus algorithm. The following are the primary research topics covered in this essay:

(1) The introduction of RabbitMQ messaging middleware will enhance the consensus communication process. The original mode of client nodes sending messages directly to the primary node is changed to the mode of client nodes sending messages to the RabbitMQ node, and the leader nodes of other groups subscribe messages from the RabbitMQ node. As a result, the most important node's communication load is reduced. The main node now operates more efficiently.

(2) It introduces the idea of grouping, where nodes are categorized into groups according to how responsive they are to the group leader node. Intra-group consensus is carried out first. Additionally, the group leader node takes part in the out-group consensus as a result of the in-group consensus, which can minimize the number of nodes engaging in the agreement and reduce communication between nodes, hence increasing the consensus's effectiveness.

(3) In order to improve the conventional PBFT three-stage consensus procedure, threshold signature is added. Instead of the original mode of net-wide broadcast messages in the preparation stage and confirmation stage, RabbitMQ nodes collect voting information and confirmation information, and perform threshold signature after verification, and then distribute it to the leader nodes of each group for confirmation. In this way, the communication times between nodes are reduced.

(4) Introduced the reputation approach and alternative group leader node mechanism, and classified the nodes into three categories based on their reputation scores: group leader nodes, candidate nodes, and common nodes. The leader node is chosen during view switching from the candidate node queue based on the credibility value, which ranges from high to low. This reduces the resource consumption on the system brought on by frequent view switching brought on by the Byzantine node being elected as an administrator node repeatedly.

2. **PBFT Consensus Algorithm.**

2.1. **Consensus Process for The PBFT Algorithm.**

Castro and Liskov [14] introduced the fault-tolerant PBFT method in 1999 to address the Byzantine general problem [16,17]. Client, the head node, and slave node are the three primary functions of the algorithm in the agreement phase. The slave node is in charge of receiving and confirming requests from the master node and other slave nodes. The client is in charge of making requests, the master node is in charge of accepting requests from the client, allocating numbers, and placing orders for final packaging transactions. All nodes communicate the agreed-upon consensus outcome to the client after it has been reached. Figure 1 depicts the consensus-building procedure.

Three steps make up the PBFT algorithm's consensus process: pre-prepare, prepare, and commit. When there are N total nodes, even if $f = (N-1)/3$ of them are defective, the system can still function normally. In Figure 1, the client node is denoted by the letter C, while the primary and slave nodes are denoted by the numbers 0 and 1-3. Three of them are faulty nodes. The PBFT consistency protocol process mainly includes:

(1) The primary node receives the message from the client.

(2) The messages are sorted, packaged, and put into fresh blocks by the master node before being broadcast to all of the network's slave nodes.

(3) Following receipt of the communication from each slave node, a simulated transaction is carried out for verification, and the new block's hash value is then determined using the transaction result and broadcast to all of the network's nodes.

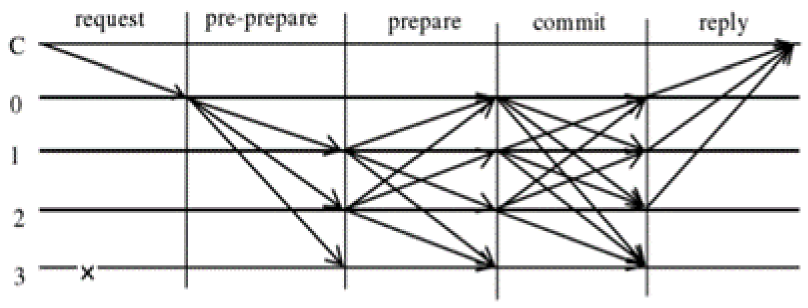

Figure 1. Process flow diagram for PBFT consensus

(4) A node broadcasts a commit message to the entire network if it gets *2f* prepare messages from other nodes.

(5) A node can commit brand-new blocks and transactions to the local blockchain and database if it gets *2f* commit messages from other nodes. And respond to the customer service end's ultimate consensus result [18].

2.2. **Security of PBFT Algorithm.** The PBFT method can tolerate less than $(N-1)/3$ malicious or flawed nodes where N is taken to be the total number of nodes in the system. The amount of standard nodes must be at least $f+1$ if you wish to maintain the system functioning correctly in the presence of f malicious nodes. The system can only establish a consensus when the overall amount of nodes is no less than $3f+1$, even in the worst-case scenario when there are f malevolent and f malfunctioning nodes in the system.

2.3. **Analysis of The PBFT Algorithm's Problems.** The PBFT algorithm still has a lot of issues even if it improves the BFT method and is commonly utilized in alliance chains [19]. Without taking into account the node's inability to achieve a three-stage consensus, the algorithm's scalability is weak. Additionally, there are significant overheads associated with network connection, and the communication difficulty between nodes is $O(n^2)$ in nature. Secondly, the primary nodes are selected sequentially, which has security risks. This selection method makes no distinction between nodes, making it simple for malicious or defective nodes to repeatedly act as primary nodes. This will result in frequent switching of views, which will negatively impact the system's ability to reach consensus and lower the system's stability and performance. Finally, since the client only communicates with the primary node, sending too many requests would negatively impact the master node's ability to handle them, lowering system throughput.

3. **TRPBFT Consensus Algorithm.**

Based on the PBFT algorithm, the TRPBFT algorithm is a reliable and effective consensus algorithm. It mainly optimizes the PBFT algorithm's three-stage consensus procedure by introducing the publish-subscribe mode of message-oriented middleware nodes and threshold signature to reduce the communication times. At the same time, through the grouping mechanism, there are fewer nodes who took part in the consensus, in order to make communication simpler. In addition, the reputation strategy and alternative group leader node mechanism are introduced, and the group leader node is chosen as the node with a good reputation. It essentially slows down the operation of the system cost caused by frequent view switching caused by the repeated selection of Byzantine nodes as group leader nodes, and improves the system operation efficiency. In Figure 2, the TRPBFT consensual model is displayed.

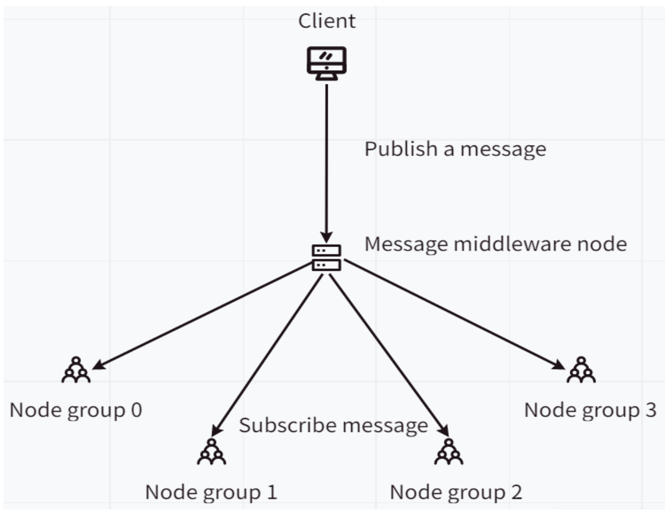

Figure 2. TRPBFT consensus model

3.1. **The Publish-Subscribe Model of Message-Oriented Middleware.** In the design of software, a release subscription is the message framework in which the sender of a message (referred to as a publisher) does not send the message directly to a specific recipient (referred to as a subscriber), but rather groups the posted messages into various categories without having to know which subscribers, if any, might exist. Similar to this, subscribers can indicate curiosity about a few or all topics to just get communications that they find interesting without knowing whether or not any publishers are available.

Figure 3 depicts the RabbitMQ publish-subscribe paradigm.

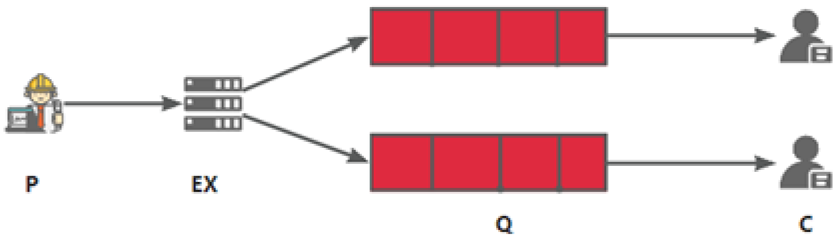

Figure 3. The publish-subscribe model in RabbitMQ

Where, P represents the producer, which is the client in the system, EX represents the switch, C represents the consumer, which is the leader of the group participating in the consensus, and the red part in the figure is the message queue.

3.2. **Threshold Signature.** Threshold signature is a kind of group-oriented cryptosystem, only some members of the group cooperate to perform encryption, decryption, signature and other operations. The (t, n) threshold signature system is one of them; it makes it more difficult to read the private key such that, even if the key of $t-1$ members is released, the system's key security can still be ensured and no legitimate threshold signature can be formed [20]. In addition, the verification of threshold signatures is very simple. Consequently, the (t,n) threshold signature technique is ideal for assuring the security of data exchange during each consensus node's process of reaching consensus in the blockchain consensus system. There are other implementation strategies for (t,n) threshold signature, but in this paper, we adopt the (t,n) threshold signature strategy based on ECDSA [21]. The specific process is as follows:

Let's assume that p and q are two extremely big prime integers, that E is an elliptic curve defined over the finite field $F(2^p)$, that P is a q basis point over E, and that both E and P are exposed.

(1) Phase-key distribution: The registry of the federated chain selects the key polynomial. And then calculate $t_i = f(i)$, and send the t_i key privately to the X_i. Calculate f_iP and $Q = dP$ and expose to all nodes, $i = 1 \sim n$. The key polynomial is shown in Formula (1):

$$f(x) = f_0 + f_1x + \cdots + f_{t-1}x^{t-1} \tag{1}$$

among them, $f_0 = d$. (2) Key verification stage: The consensus node determines if the key is a point on the curve of ellipsis E by calculating the value of the verification formula to validate its validity. The verification formula is shown in Formula (2):

$$t_iP = \sum_{j=0}^{t-1} i^j(f_iP) \tag{2}$$

(3) Signature phase: Each signature node X_i randomly selects a number k_i. Calculate $R_i = k_iP$. And open to all signatories. After each signature node X_i calculate the values of (x_i, y_i), e_i, Q_i, r and s_i respectively according to Formula (3)-(7). To the node that gathered the signature, deliver the result set (r, s_i, Q_i). The node that obtains the signature then checks to see if the R_i of each node X_i that has been signed is equal to $s_iP - rQ_i$. The node that gathers the signature determines S's value using Formula (8) and furthermore sends the group signature (m, r, S) to the verification node for validation if the verification is successful.

$$(x_i, y_i) = \sum_{i=0}^{t-1} R_i \tag{3}$$

$$e_i = a_it_i, \quad \text{among} \quad \left(a_i = \frac{\prod_{h\neq i} h}{h-i}\right) \tag{4}$$

$$Q_i = e_iP \tag{5}$$

$$r = x_i - h(m) \mod q \tag{6}$$

$$s_i = e_ir + k_i \mod q \tag{7}$$

$$S = \sum_{i=0}^{t-1} s_i \mod q \tag{8}$$

(4) Verification stage: Each verification node determines whether $SP-rQ$ is valid when $x=x_i$ by calculating the value of $SP-rQ$ as point (x,y) to verify whether the signature is valid.

3.3. **Grouping Mechanism.** This work offers a system on the basis of grouping the quick response rate of the group leader node in order to reduce communication time and increase communication efficiency between nodes. The specific steps are as follows:

(1) With the use of the alliance chain's access mechanism, n nodes are randomly chosen as the node of leadership when a node joins the alliance chain. The other nodes are then grouped based on the leader node's response time, and the nodes are then split into G consensus groups.

(2) The group leader node j maintains a list of group members L, so that it can decide whether to allow new nodes to join the group. The captain of the group node can broadcast a message that enable new members to be received $\langle GROUP, t_1, L\rangle > \delta_i$ when a particular amount of nodes in a group is less than the maximum number of nodes N. The group leader node's signature is represented by δ_i, and t_1 is the timer.

(3) When the authentication is successful, a node sends the group leader a broadcast message, followed by an application message, using the format $\langle GROUP-ADD, x, t_2\rangle > \delta_x$. Where δ_x is the signature of node x.

(4) The leader node will verify after receiving the application information of other nodes. Once the check is successful, the node will be included in the member list L of the group and send the confirmation information to it $\langle GROUP-ACK, t_3, L\rangle > \delta_i$.

(5) The group leader node transmits its member list to the other group leader nodes after the group is finished. For the process of consensus, the group leader node delivers the member list to the group members after confirming the member list.

As the node representing the group at the global consensus, the leader node represents the group and has among the greatest credits score in the network. A node's credit value will vary after a consensus round. As a result, the group leader node should now be chosen again in accordance with the node's new credit value, and the previous procedure should be re-grouped.

3.4. **Reputation Strategy and Alternative Group Leader Node Mechanism.** In the PBFT algorithm, when performing view switching, the formula p=(v+1) mod N is used to determine the order in which the principal node is chosen. This selection method has certain security risks. It is simple to choose Byzantine nodes as master nodes repeatedly that causes frequent switching of views and wastes system resources, affecting the performance and security of the system [22]. To this end, this paper introduces the reputation strategy and alternative leader node mechanism, through the evaluation of the credibility value of nodes, it is divided into: leader node, candidate contact and common node. Therefore, the selection of group leader nodes is optimized, so that nodes with high reputation have more opportunities to act as group leader nodes. Consequently, the system's vulnerability to Byzantine node damage is diminished, and the system's overall performance and security are enhanced. This study evaluates the node's current reputation value based on past reputation value, other assessment indicators, and if the node's voting information is compatible with the consensus outcome [23,24,25].

Definition 3.4.1. The nodes' performance in the most recent consensus process is taken into greater consideration when calculating a node's reputation value thanks to the time decay mechanism. Its Formula is as follows (9):

$$T_i = e^{-|t_{now}-t_{bef}|} \tag{9}$$

Where, T_i represents the attenuation degree of the consensus performance of node i with the change of time, t_{now} represents the time of the current consensus, and t_{bef} represents the time of the last consensus.

Definition 3.4.2. The reliable value of reaching a transaction consensus refers to whether the voting information of each node is consistent with the final consensus result [18]. Its formula is as follows (10):

$$R_i(X) = \begin{cases} T_i \times R_i(X-1) + 1, & \text{The node successfully participated in a consensus} \\ \frac{1}{3} \times R_i(X-1), & \text{The node timed out and did not respond} \\ 0, & \text{The node is detected to be doing evil} \end{cases} \tag{10}$$

Where X is the frequency of consensus participation by node i, $R_i(X)$ represents the reliable value after node participates in consensus X times, and the initial value of $R_i(X)$ is 0.

Definition 3.4.3. The node's previous credit value [19]. Its formula is as follows (11):

$$C(i)' = yC(i-1) \tag{11}$$

where y stands for the previous state's influencing element.

Definition 3.4.4. The following formula (12) represents the node's final credit value:

$$C(i) = \frac{1}{2}\left(x \times \frac{R_i(X)}{\max(R(X))} \times 100 + y \times C(i-1)\right) \tag{12}$$

where x is the weight of the reliable value to reach the transaction consensus, $x + y = 1$. $\max(R(X))$ is the reliability value of the best performing node in the consensus.

3.5. **Consensus Flow of TRPBFT Improved Algorithm.** The TRPBFT consensus algorithm can be divided into in-group and out-of-group consensus. When the leader node collects enough vote confirmation information, out-of-group consensus will be enabled. Figure 4 depicts the TRPBFT consensus algorithm's consensus procedure. Below is a description of the specific consensus process:

Requesting stage: The client communicates with the RabbitMQ node by sending a request message.

Pre-Preparation stage: The $\langle \text{pre-prepare}, h, n, t, v, \text{Dig}(m)\rangle \delta_i$ message is broadcast to the member nodes of each group by the leader of each group after subscribing to a message from the RabbitMQ node. Where $\text{Dig}(m)$ is the message digest, h is the block height, n is the number of the serial number of the present request, t is a time stamp, v is the perspective number, and δ_i is the group leader node's signature.

Preparing stage: Each group's leader node gathered and validated the member nodes' verification data as part of the preparation process $\langle \text{prepare}, h, n, t, v, \text{Dig}(m)\rangle \delta_x$. The out-of-group consensus is enabled and the outcome of the intra-group consensus is sent to the RabbitMQ node after a sufficient number of votes have been gathered. The RabbitMQ node collects and verifies the voting information, and after that, sends each group's leader node the threshold signature for verification. δ_x indicates the signature of the x member node.

Confirmation stage: The leader of each group sends the confirmation information $\langle \text{commit}, h, n, t, v, \text{Dig}(m)\rangle \delta_i$ to the RabbitMQ node. The RabbitMQ node collects, verifies and collects statistics. When sufficient confirmation information has been collected, the RabbitMQ node then issues the consensus message $\langle \text{commit}, h, n, t, v, \text{Dig}(m)\rangle \delta_r$ to all group leaders. The consensus message is broadcast to the group's members by the leader of each group after they have received it $\langle \text{commit}, h, n, t, v, \text{Dig}(m) \rangle \delta_i$. The RabbitMQ node's signature is denoted by δ_r.

Reply phase: The RabbitMQ node replies with a message that the client nodes have reached a consensus.

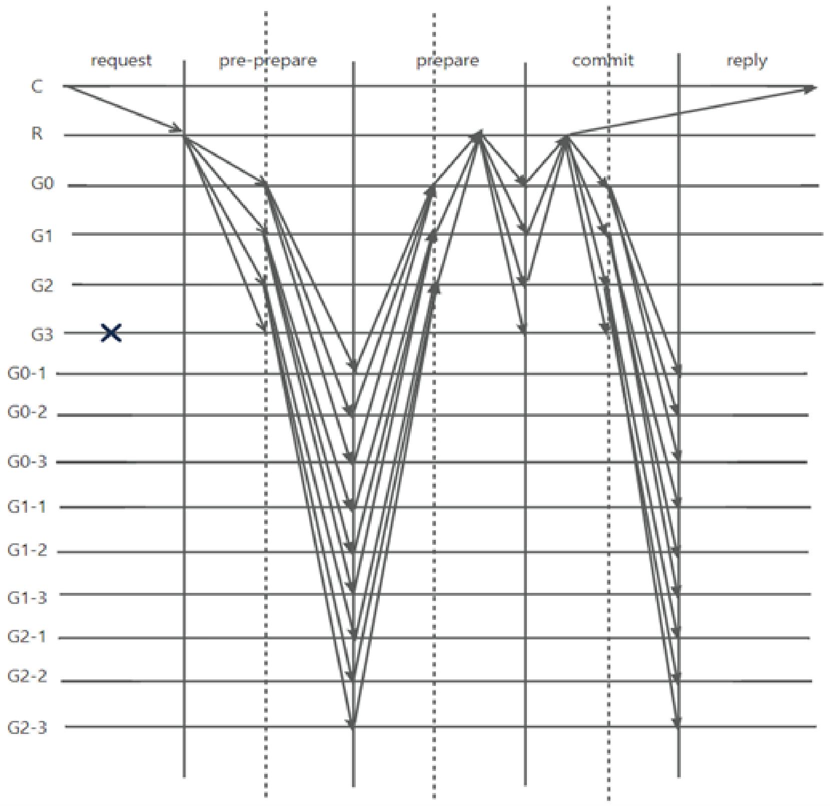

Figure 4. TRPBFT consensus process

4. **Examination of Experiments and Results.**

4.1. **Environment for Experimentation.**

The revised TRPBFT method is developed using the high-level programming language Golang, and the experiment of simulation is carried out using Hyperledger Fabric v1.4, in order to demonstrate the effectiveness of the approach.

The experiment's hardware setup is as follows:
(1) CPU Intel(R) Core(TM) i5-11320H @ 3.20GHz 2.50GHz,
(2) Memory 16GB,
(3) Ubuntu 22.04.2(LTS) 64-bit OS,
The software environment is:
(1) GoLand2023.1.2,
(2) Hyperledger Fabric v1.4,
(3) Docker v20.10.9, Go v1.20.3.

In the experiment, communication costs, tolerance for failure, throughput, and consensus time were examined between 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, and 60 nodes using the PBFT consensus method. The client transmitted 150 transaction messages each time the experiment was run, yielding an average value of 800 runs, which served as the experiment's output.

4.2. **Cost of Communication.** When a system's nodes communicate with one another to reach a consensus, this is referred to as communication overhead. Assume that N nodes make up the entire system. The pre-preparation, the platform, preparation phase, as well as confirmation state are where nodes communicate the most when using the PBFT algorithm. The central node will hear from the client while the phase of preparation is underway. All slave nodes are notified by a network-wide broadcast, with communication times of $N-1$. During the drafting stage, all the slave nodes broadcast their voting results across the network after executing the verification transaction, and there are currently $(N-1)^2$ communications. After obtaining consensus, all nodes will publish the last verification data across the network's resources during the confirmation stage. There have been $N(N-1)$ communications as of right now. In conclusion, the three steps of the PBFT algorithm's total communication frequency (T_1) are as follows:

$$T_1 = N - 1 + (N-1)^2 + N(N-1) = 2N(N-1) \tag{13}$$

This study proposes an enhanced TRPBFT method and assumes that the system's N nodes are organized into M groups. Each group's leader subscribes to messages from a RabbitMQ node during the setup phase and broadcasts the messages to every node in the group. The transmission intervals are $M + M(N/M - 1)$. The intra-group node conducts and validates the transaction during the intra-group preparation phase, then sends the validated vote result to the group leader node for collection and counting. There are $M(N/M - 1)$ communications. The group leader node starts the out-group consensus and delivers the decision results to the RabbitMQ node once it has accumulated enough messages to show that it has passed the vote. The vote results are gathered and verified by the RabbitMQ node. After the verification succeeds, the RabbitMQ node signs the vote result and sends the signed result to all the group leader nodes for verification. In this case, there are $2M$ communications. The terrorist organization leader delivers the verification result to the RabbitMQ nodes for confirmation at the conclusion of the verification stage. The RabbitMQ node then communicates the outcome of the final consensus to each group leader. The group leader communicates the final consensus decision to the other group leaders after receiving the message. There have been $M + M + M(N/M - 1)$ communications. In conclusion, the TRPBFT algorithm's four phases' total communication frequency (T_2) are as follows:

$$T_2 = M + M\left(\frac{N}{M} - 1\right) + M\left(\frac{N}{M} - 1\right) + 2M + 2M + M\left(\frac{N}{M} - 1\right) = 3N + 2M \tag{14}$$

Let us say there are N nodes in the system, and there are four groups. Figure 5 compares the experimental findings of communication times between PBFT and TRPBFT algorithms for various node totals.

As demonstrated in Figure 5, the PBFT and TRPBFT algorithms both require longer communication durations to attain consensus as the total number of nodes grows. While TRPBFT algorithm communication frequency increases slowly, PBFT algorithm communication frequency rises quickly. Compared to the PBFT method, the TRPBFT algorithm has a lower communication frequency. Conclusion: When compared to the PBFT method,

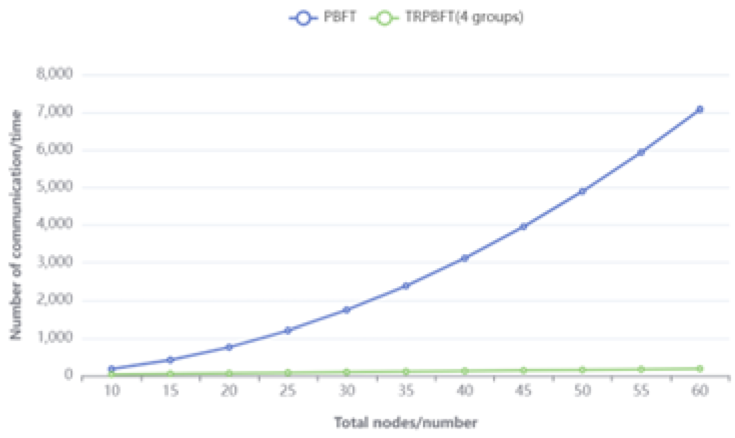

Figure 5. Comparison of communication times of PBFT and TRPBFT

the TRPBFT approach can significantly shorten the amount of time that is spent communicating during the consensus process.

4.3. **Tolerance for Faults.** TRPBFT consensus algorithm is an optimization of PBFT consensus algorithm, but it changes the communication process. The underlying fault tolerance system of PBFT is the same as that of this system. Because of this, the fault tolerance for PBFT and out-of-group consensus is the same. However, the TRPBFT consensus algorithm introduces a grouping and reputation strategy, so that more malicious nodes and faulty nodes can be accommodated within the group. When all nodes in the group are malicious or faulty nodes, the leader node participates in the out-of-group consensus as a Byzantine node; when less than half of the nodes in the group are malicious or faulty nodes, the leader node participates in the out-of-group consensus as a non-Byzantine node. Therefore, 1/3 group is the total of all malicious nodes and defective nodes, and 2/3 group is the total of less than half of the nodes in the group. These numbers represent the greatest number of malicious nodes or faulty nodes that the TRPBFT consensus method can tolerate. Let us assume that there are N total nodes in the system, separated into M groups. The TRPBFT algorithm may support a maximum of the following Byzantine nodes E:

$$E = \frac{N}{M} \times \frac{M}{3} + \frac{2M}{3} \times \left(\frac{1}{2} \times \frac{N}{M} - 1\right) = \frac{2}{3}(N - M) \tag{15}$$

Assume the system's N nodes are split into four groups. Figure 6 displays the comparison experiment findings for the maximum Byzantine nodes that PBFT and TRPBFT algorithms can support under various node totals.

Figure 6 shows that as the overall number of nodes rises, the greatest amount of Byzantine nodes that PBFT and TRPBFT algorithms can support likewise rises. The TRPBFT algorithm can support a greater number of Byzantine nodes than the PBFT method can, always. It can be concluded that TRPBFT algorithm has higher fault tolerance and security under the same conditions.

4.4. **Throughput.** In blockchain systems, throughput is a crucial metric for determining the system's capacity for processing transactions. It refers to the quantity of messages the system processes per unit of time, typically denoted in *TPS*. The following (16) is the calculating Formula for it:

$$TPS = \frac{Transactions}{\Delta t} \tag{16}$$

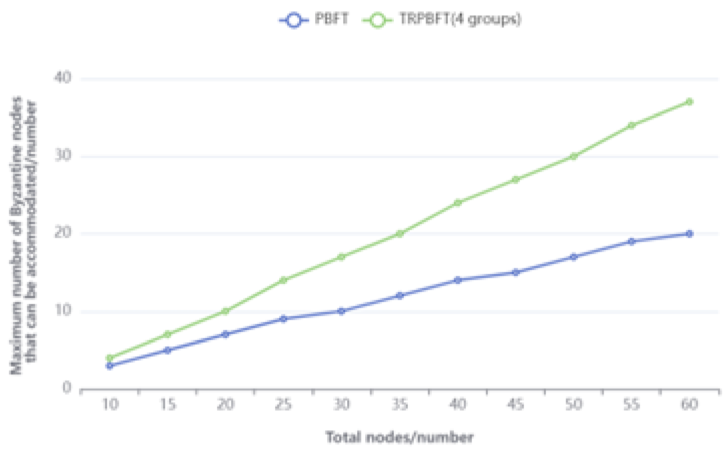

Figure 6. The greatest number of Byzantine nodes that PBFT and TRPBFT can support is compared

Transactions indicates the total number of messages processed in a period of Δt.

If the system's N nodes are split into four groups, then Figure 7 compares the experimental findings of the transaction throughput of the PBFT and TRPBFT algorithms with various node counts. Figure 7 shows that the transaction throughput of the PBFT

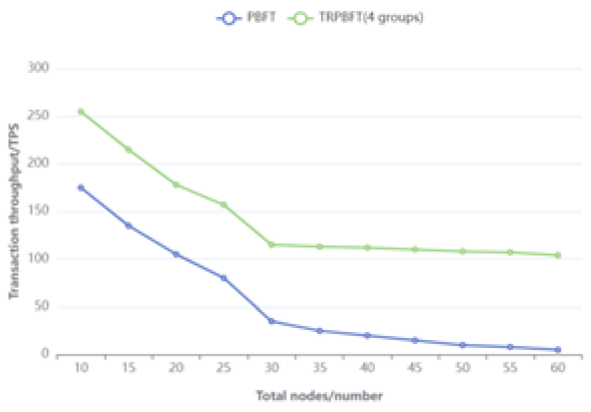

Figure 7. Comparison of transaction throughput between PBFT and TRPBFT

and TRPBFT algorithms rapidly declines as the total number of nodes increases. The TRPBFT algorithm consistently has a higher transaction throughput than the PBFT method. While the transaction throughput of TRPBFT declines gradually as the number of nodes rises, the transaction throughput of PBFT drops off quickly. This result is due to the introduction of RabbitMQ node for asynchronous communication and its publish-subscribe mode as well as the three-stage PBFT algorithm consensus process's usage of a threshold signature to improve it. Conclusion: The TRPBFT algorithm outperforms the PBFT method in terms of processing volume of transactions simultaneously, significantly enhancing the system's ability to reach consensus.

4.5. **Consensus Delay.** Consensus delay is the amount of time between the start of a transaction package becoming a block and the moment when all nodes in the system have reached a consensus and produced a block. The consistent protocol's running speed can be determined using this crucial index. The security and operational efficiency of the system can all be improved along with the practicality of the system by reducing the consensus delay. The system can also be made more secure and efficient by reducing the consensus delay. Its calculation Formula is as follows (17):

$$T_d = T_c - T_g \tag{17}$$

Among them, T_d represents consensus delay, T_c represents transaction confirmation time, and T_g represents transaction generation time.

Let us say the system's N nodes are split into four groups. Figure 8 displays the comparison experiment findings for the consensus latency between the PBFT and TRPBFT algorithms under various node totals.

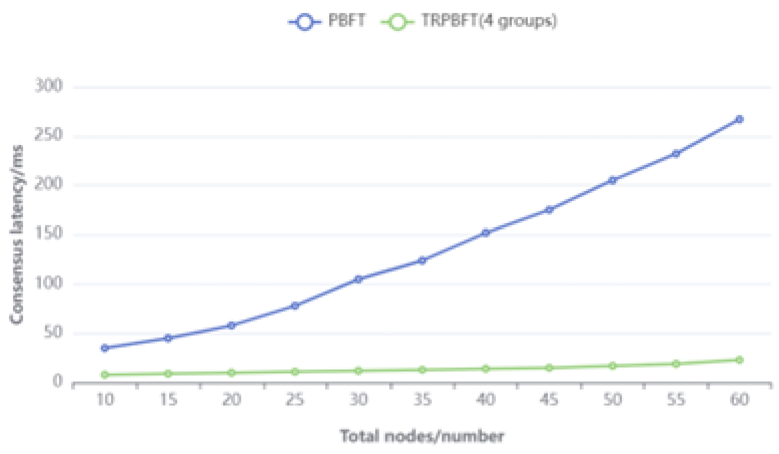

Figure 8. Consensus delay comparison between PBFT and TRPBFT

Figure 8 illustrates how the consensus delay of the PBFT and TRPBFT algorithms both rise when the system's overall number of consensus nodes does. With more nodes in the system, the consensus delay of the PBFT algorithm increases quickly, whereas the consensus delay of the TRPBFT method increases more slowly. Compared to the PBFT algorithm, the TRPBFT algorithm has a shorter consensus delay. The TRPBFT algorithm uses the grouping and reputation mechanism, as well as the threshold signature, to optimize the three-stage consensus process, which reduces the number of participating nodes and the communication times between nodes. Conclusion: When compared to the PBFT method, the TRPBFT approach has superior scalability and stability and is more appropriate for alliance chain architectures with many nodes being involved in agreement.

5. **Conclusion.** There are a number of issues with the PBFT consensus algorithm, including inadequate scalability, high network transmission costs, and security vulnerabilities in the choice of master nodes. The TRPBFT consensus algorithm, which is based on RabbitMQ and threshold signature, is presented in this study. The algorithm introduces the grouping mechanism, message-middleware RabbitMQ and threshold signature to optimize the original three-stage PBFT consensus process, making the original pin-to-pin-communication between the whole network nodes in the preparation stage and the confirmation stage. The vote information and confirmation information are collected by the RabbitMQ nodes for verification, and the threshold signature is then sent to the group leader for verification. The network's scalability and communication effectiveness are significantly increased as its transmission complexities is decreased from the initial O(n2) to O(n). In order to improve the selection of group leader nodes and reduce the overhead of frequent view switching brought on by the repeated selection of group leader nodes by Byzantine nodes, the reputation strategy and alternative group leader node

mechanism are introduced at the same time. According to experimental findings, the TRPBFT algorithm significantly enhances the performance and dependability of the system when compared to the PBFT algorithm in terms of fault tolerance, communication costs, consensus delay, throughput, and other factors. However, the algorithm still has some limitations in terms of scalability and security. For example, although the algorithm makes it possible to accommodate more Byzantine nodes in the system by using a grouping strategy, the algorithm does not take into account the dynamic joining and exiting of nodes during the operation of the system. Also the algorithm reduces the network traffic and reduces the access pressure on the master node by introducing the publish-subscribe mechanism and threshold signatures of RabbitMQ nodes, but the algorithm does not take into account the problem of RabbitMQ nodes that cannot work properly if they are attacked. In this regard, in the future work, we will further study the problem of allowing nodes to dynamically join and exit the algorithm as well as how to ensure that the system can still operate normally when the RabbitMQ nodes are attacked, in order to further optimize the performance of the algorithm and apply the algorithm in practical applications to solve the problem of poor scalability of the federation chain.

Acknowledgement. The research is supported by: Heilongjiang Provincial Higher Education Institutions Basic Scientific Research Business Fund Project (2022-KYYWF-0532).

REFERENCES

[1] G.-Q. Liang, S.-R. Weller, J.-H. Zhao, F.-J. Luo, and Z.-Y. Dong, "The 2015 Ukraine Blackout: Implications for False Data Injection Attacks," IEEE Transactions on Power Systems, vol. 32, no. 99, pp. 3317-3318, 2017.

[2] J. Jiang, and Y. Qian, "Defense Mechanisms against Data Injection Attacks in Smart Grid Networks," IEEE Communications Magazine, vol. 55, no. 10, pp. 76-82, 2017.

[3] R.-R. Jiang, Z.-Q. Weng, and T.-M. Chen, "Development of Industrial Internet platform and its security technology," Telecommunications Science, vol. 36, no. 3, pp. 3-10, 2020.

[4] M.S. Ali, M. Vecchio, and M. Pincheira, "Applications of block-chains in the Internet of Things: a comprehensive survey," IEEE Communications Surveys and Tutorials, vol. 21, no. 2, pp. 1676-1717, 2019.

[5] J.-Q. Huang, L.-H. Kong, and G.-H. Chen, "Towards secure in-dustrial IoT: blockchain system with credit-based consensus mechanism," IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3680-3689, 2019.

[6] Z. Zheng, S. Xie, and H.-N. Dai, "Blockchain challenges and opportunities: A survey," International Journal of Web and Grid Services, vol. 14, no. 4, pp. 352-375, 2018.

[7] J. Wan, J. Li, and M. Imran, "A Blockchain-Based Solution for Enhancing Security and Privacy in Smart Factory," IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3652–3660, 2019.

[8] C.-M. Chen, X.-T. Deng, S. Kumar, S. Kumari, and S.-K. Islam, "Blockchain-based medical data sharing schedule guaranteeing security of individual entities," Journal of Ambient Intelligence and Humanized Computing, 2021. [Online]. Available: https://doi.org/10.1007/s12652-021-03448-7.

[9] J.-H. Chen, H. Xiao, M.-C. Hu, and C.-M. Chen, "A blockchain-based signature exchange protocol for metaverse," Future Generation Computer Systems, vol. 142, pp. 237-247, 2023.

[10] C. Sun, "Research on Power grid data security based on blockchain technology," Nanjing University of Posts and Telecommunications, 2021.

[11] C.-C. Dai, H.-J. Luan, and X.-Y. Yang, "Research review of blockchain technology," Computer Science, vol. 48, no. 11A, pp. 500-508.

[12] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," Concurrency: the Works of Leslie Lamport, 2019, pp. 203-226.

[13] L. Lamport, "The Part-Time Parliament," ACM Transactions on Computer Systems, vol. 16, no. 2, pp. 133-169, 1998.

[14] M. Castro, and B. Liskov, "Practical byzantine fault tolerance," Proceedings of the 1999 3rd Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 1999, pp. 173-186.

[15] D. Ongaro, and J. Ousterhout, “In search of an understandable consensus algorithm,” 2014 USENIX Annual Technical Conference (Usenix ATC 14). 2014, pp. 305-319.

[16] M. Pease, R. Shostak, and L. Lamport, “Reaching agreement in the presence of faults,” Journal of the ACM, vol. 2, no. 4, pp. 228-234, 1980.

[17] J. Bruna, W. Zaremba, and A. Szlam, “Spectral networks and locally connected networks on graphs,” arXiv preprint, 2014. [Online]. Available: https://arxiv.org/abs/1312.6203.

[18] X.-T. Wei, “Analysis and improvement of PBFT consensus mechanism in Fabric,” Dalian Maritime University, 2019.

[19] S.-N. Liu, R.-H. Zhang, and C.-Z. Liu, “An improved PBFT consensus algorithm based on grouping and credit rating,” Computer Engineering, pp. 1-13, 2023.

[20] J. Ning, “Research on (t, n) Threshold digital signature technology based on Elliptic curve,” Guangxi University, 2006.

[21] J.-R. Han, J.-Q. Lv, and X.-M. Wang, “Verifiable Threshold Signature Scheme Based on Elliptic Curve,” Journal of Xidian University (Natural Science Edition), vol. 30, no. 1, pp. 26-28, 2003.

[22] G. Xu, H. Bai, and J. Xing, “SG-PBFT: A secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent Internet of vehicles,” Journal of Parallel and Distributed Computing, vol. 164, pp. 1-11, 2022.

[23] Y. Chen, M. Li, and X. Zhu, “An improved algorithm for practical byzantine fault tolerance to large-scale consortium chain,” Information Processing and Management, vol. 59, 102884, 2022.

[24] J.-H. Chen, X. Zhang, and P.-F. Shangguan, “Improved PBFT algorithm based on reputation and voting mechanism,” IOP Publishing, 2020.

[25] P. Liu, S. Ren, and J. Wang, “A Blockchain Consensus Optimization-Based Algorithm for Food Traceability,” Mobile Information Systems, 2022.