Hindawi
Security and Communication Networks
Volume 2022, Article ID 8311821, 11 pages
https://doi.org/10.1155/2022/8311821

Research Article

GPBFT: A Practical Byzantine Fault-Tolerant Consensus Algorithm Based on Dual Administrator Short Group Signatures

Xiaosheng Yu[1,2], **Jie Qin**[1,2] **and Peng Chen**[1,2]

[1]*Hubei Province Engineering Technology Research Center for Construction Quality Testing Equipments, China Three Gorges University, Yichang 443002, China*
[2]*College of Computer and Information Technology, China Three Gorges University, Yichang 443002, China*

Correspondence should be addressed to Xiaosheng Yu; yuxiaosheng@ctgu.edu.cn

Received 13 April 2022; Accepted 14 July 2022; Published 5 August 2022

Academic Editor: Jiewu Leng

The practical Byzantine fault-tolerant consensus algorithm reduces the operational complexity of Byzantine protocols from an exponential level to a polynomial level, which makes it possible to apply Byzantine protocols in distributed systems. However, it still has some problems, such as high communication overhead, low security, poor scalability, and difficulty in tracking. In this article, we propose a Byzantine fault-tolerant consensus algorithm based on dual administrator short group signatures (GPBFT). Firstly, the certification authority chooses the master node and group administrators based on the credit value. The group administrators organize the nodes into a group, and the members generate the signatures by applying the short group signatures scheme, in which any group member can represent the group during the GroupSign phase. Additionally, the GPBFT algorithm adds the Trace phase. According to member and client authentication information, the group administrator can track the true identity of the malicious node, identify the malicious node, and revoke it. The experimental results show that compared with the PBFT algorithm, the GPBFT algorithm can reduce the network communication overhead, reduce the consensus delay, and greatly improve the security and stability of the system. The algorithm can effectively manage member nodes and enable the tracking of identified malicious nodes while maintaining anonymity in terms of node tracking.

1. Introduction

In the practical applications of blockchain, storage scalability and security are the major problems that researchers confront in the existing field of sustainable manufacturing. The primary security problems include the generation and protection of private keys, vulnerabilities of the signature algorithm, the centralization of the consensus process, vulnerabilities of smart contracts, and vulnerabilities of decentralized applications. It is challenging to design scalable and highly secure consensus algorithms to assist self-adaptive coordination effectively in each sustainable manufacturing system [1, 2]. The consensus algorithm is the core mechanism in the blockchain system, and it aims at solving the problem of data consistency across distributed nodes in the system [3–5]. The Byzantine fault-tolerant algorithm (BFT) is a fault-tolerant algorithm based on the Byzantine problem, which addresses how to reach consensus with reliable communication but the possibility of node failure [6]. However, the algorithm's exponential operational complexity makes it difficult to implement in practice. In Ref. [7], Castro and Liskov proposed the PBFT algorithm, an improved algorithm of BFT, which reduces the operational complexity of Byzantine protocols from the exponential level to the polynomial level, allowing Byzantine protocols to be used in distributed systems. The Hyperledger Fabric project was the first to use the PBFT algorithm in the consortium blockchain [8–10]. The Tendermint algorithm of the Cosmos blockchain combines the PBFT and the PoS algorithm and uses a token mortgage to select some consensus nodes for BFT consensus. It weakens the asynchronous assumption and incorporates the concept of lock based on the PBFT algorithm, allowing consensus nodes to reach consensus through two-stage communication in a partially

synchronous network [11, 12]. Based on Tendermint, the Hotstuff algorithm integrates the blockchain's chained-block structure with each phase of BFT, where the signatures confirmation of the previous block and the construction of a new block are performed simultaneously between nodes at each phase, simplifying the algorithm's implementation [13–15]. The MBFT algorithm combines hierarchical and slicing technology. The former can reduce the load of individual nodes and effectively improve consensus efficiency. The latter can assign transactions to different node groups to improve the processing power and decrease delay [16]. By grouping the network nodes, RBFT adopts the improved RAFT to participate in the consensus within the group. The leaders generated by the RAFT algorithm form a new group, and the PBFT consensus mechanism is adopted among the new groups. The algorithm solves the problem that some traditional PBFTs cannot support low delay, high throughput, and high security in large-scale networks [17]. The above consensus algorithms primarily use a subset of nodes to replace the entire network, which can reduce the traffic and improve the algorithm efficiency. However, in large-scale networks, only a few nodes participating in the consensus will have an impact on the system's degree of centralization, and the scalability is limited. On the other hand, many improvements of the algorithm are based on grouping or hierarchical thinking. Although the traffic of the PBFT algorithm can be reduced by dividing the consensus process into multiple levels, the algorithm still maintains a high complexity. Furthermore, the existing improved algorithms do not put the security of the PBFT algorithm in the first place or improve the handling of malicious nodes.

In this article, we propose a practical Byzantine fault-tolerant consensus algorithm based on dual administrator short group signatures, in which the client initiates a request to start the consensus process after selecting the master node and the group administrator with short group signatures. Compared with the PBFT algorithm, the GPBFT algorithm adds the GroupSign phase and Trace phase. In the GroupSign phase, the group administrator organizes the replica nodes into a group, in which one group member can represent the group as long as it completes the consensus process, which will reduce the communication overhead and decrease the number of communications. In the Trace phase, the group tracking administrator can trace the specific identity information of the node whose authenticator failed to verify and then revoke the member to ensure the security and stability of the system.

2. Related Work

Chaum and van Heyst introduced the concept of group signatures in 1991 [18]. Camenish et al. later modified and refined the concept [19, 20]. Group signatures are widely used in management, military, political, and economic aspects. Group signatures, like other digital signatures, can be verified publicly and only with a single group public key. The group administrator in a group signature ensures that the signature is secure and traceable, in addition to basic anonymity. The group administrator can search for the real signer by opening the group signatures [21].

2.1. The Foundation of Short Group Signatures. Boneh, a professor at Stanford University, proposed short group signatures for the first time at the International Conference on Cryptography in 2004 [22]. The security of this signatures scheme is based on the strong Diffie–Hellman (SDH) and linear Diffie–Hellman (LDH) assumptions in cryptography. The signatures use bilinear mapping e: $G_1 \times G_2 \longrightarrow G_T$, which guarantees the length of the signature while satisfying the characteristics of the group signatures and meets the security criteria.

2.1.1. Bilinear Mapping, and SDH and LDH Assumptions

Bilinear mapping: Let G_1, G_2, and G_T be three multiplicative cyclic groups of prime order n, and the generating element of G_n is g_n. A bilinear mapping is a mapping relation e: $G_1 \times G_2 \longrightarrow G_T$ defined on these three groups, satisfying bilinearity, nondegeneracy, and computability.

q-Strong Diffie–Hellman ($q - S\ DH$): Given $(q+2)$tuples $(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, \ldots, g_2^{\gamma^q})$ as input and a pair of $(g_1^{1/(\gamma+x)}, x), x \in Z_p^*$ as output.

Linear Diffie–Hellman (LDH): Decision linear problem in G_1: $u, v, u^a, v^b, h^c \in G_1$ are given as input, and the output is Yes if $a + b = c$; otherwise, the output is No.

2.1.2. Short Group Signatures Technology. In a short group signature scheme, any member of a group can sign messages anonymously on behalf of the entire group. Short group signatures, like all other digital signatures, are publicly verifiable and can be verified with just one group public key, as shown in Figure 1.

2.1.3. Short Group Signatures Security Standards. Assuming that communication between the group members and administrators is confidential, a short group signature scheme should ensure that the signature system is both effective and long-lasting. The properties it needs to meet are shown in Table 1.

2.2. The PBFT Consensus Algorithm. The practical Byzantine fault-tolerant consensus algorithm is a distributed consistency algorithm based on state machine replication. It requires each node to sign when sending messages, and other nodes cannot modify other nodes' messages. After receiving a client request, the next request will be sent for execution only after the completion of the previous request by network-wide broadcast.

In the PBFT algorithm, all nodes operate in the same configuration, where there is only one master node and the other nodes act as replica nodes. The master node is responsible for sorting the requests from the clients and sending them to the replica nodes in order. The basic process of the whole algorithm is shown in Figure 2.

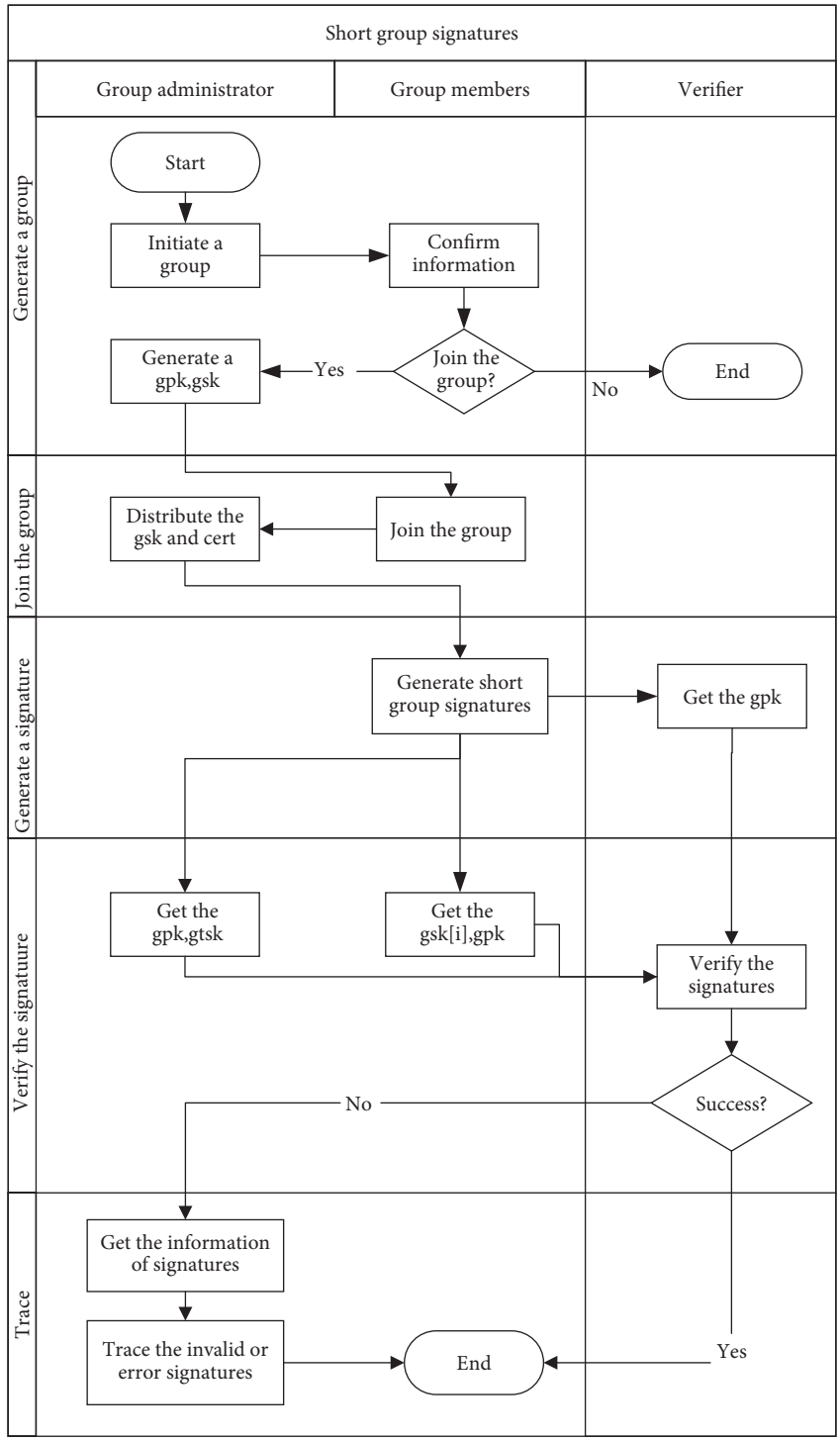

Figure 1: The short group signatures swimlane flowcharts.

There are three core phases in the process of the PBFT algorithm: the Pre-prepare phase, the Prepare phase, and the Commit phase. At first, a client sends a request to the master node. Then, the master node $N0$ will send a Pre-prepare message to the other nodes after receiving the client request. Other nodes start the core three-phase consensus process after receiving the Pre-prepare message. The details are as follows:

(a) Pre-prepare Phase: The node decides whether to agree to the request based on the message content or the request number order after receiving the Pre-Prepare message.

(b) Prepare Phase: After agreeing to the request, the node sends a prepare message to other nodes. If more than $2f$ (f denotes the maximum number

Table 1: Short group signatures security standards.

Security standards	Explanation
Correctness	Legal group members' signature is properly verified and that the group signature can be traced back to the original signer
Unforgeability	A legal group signature can only be generated by members who have obtained a group membership certificate and a signing key
Anonymity	The user who receives the signature can only verify the signature's legality, not the identity of the group member who generated it, or even the identity of the other members in the group
Traceability	Only the administrator can open a signature and find the identity of a signed group member
Unlinkability	It is computationally impossible to determine whether two signatures are signed by the same group member for unopened signatures
Irreplaceability	No member of the group can generate a signature on behalf of other users
Anti-joint attack	Even if some group members are federated, they cannot produce a valid group signature that can be tracked by the group administrator

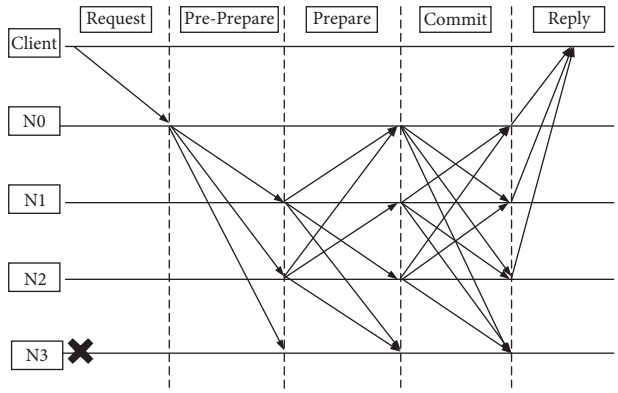

Figure 2: The PBFT consensus process.

of fault-tolerant malicious nodes) different nodes receive a prepare message within a certain time, the Prepare phase is complete.

(c) Commit Phase: Broadcast commit messages to other nodes. When $2f + 1$ commit messages are received (including its own), most of the nodes have entered the Commit phase, and consensus has been reached in this phase, so the node executes the request and writes the data.

The node sends a message to the client when the process is finished.

3. The Dual Administrator Short Group Signatures Scheme

When designing a group signature scheme, the length of a group signature has always been an important factor. When the network bandwidth is limited, short group signatures are commonly used. The short group signatures can guarantee the group member's privacy, and one of the main advantages of this scheme is that the signature is short. For example, when the elements in G_1 are 171-bit strings, the signature's length is only 192 bytes, which can reduce the system's communication load. Moreover, the security is approximately the same as that of the RSA signature algorithm with a signature length of 1024 bits. To ensure the stability and security of the signature algorithm, the dual administrator short group signatures scheme in this article must satisfy the following three conditions:

(a) A group signature scheme may require the membership revocation to simplify the membership management.

(b) Given the limited storage resources of the blockchain, the signature data cannot be too large.

(c) The group membership administrator initiates requests to establish groups, select users with high reputation as the group members, and select the group tracking administrator who is responsible for determining the members, opening the group signatures, and tracking the malicious users.

The dual administrator short group signatures scheme is shown in Figure 3.

The process is as follows:

(a) Initialize

Initialize (n): Initialize algorithm, that is, two group administrators establish a group according to the relevant parameters. The input parameter is n, where n is the number of the group members (including the administrators). The outputs are the group public key gpk, the group tracking private key $gtsk$, and the group member's private key $gsk[i]$.

(b) Join

Join (x_i): Join algorithm, that is, the process of user $i (1 \le i \le n)$ applying to join the group. The input parameter is x_i. The group member i randomly selects $x_i \in Z_p^*$ as the user's private key. The output is the group member's private key $gsk[i]$ of user i.

(c) Sign

Sign ($gpk, gsk[i], M$): The signature algorithm, that is, the process of group signatures of the message M by the group members. The inputs are the group

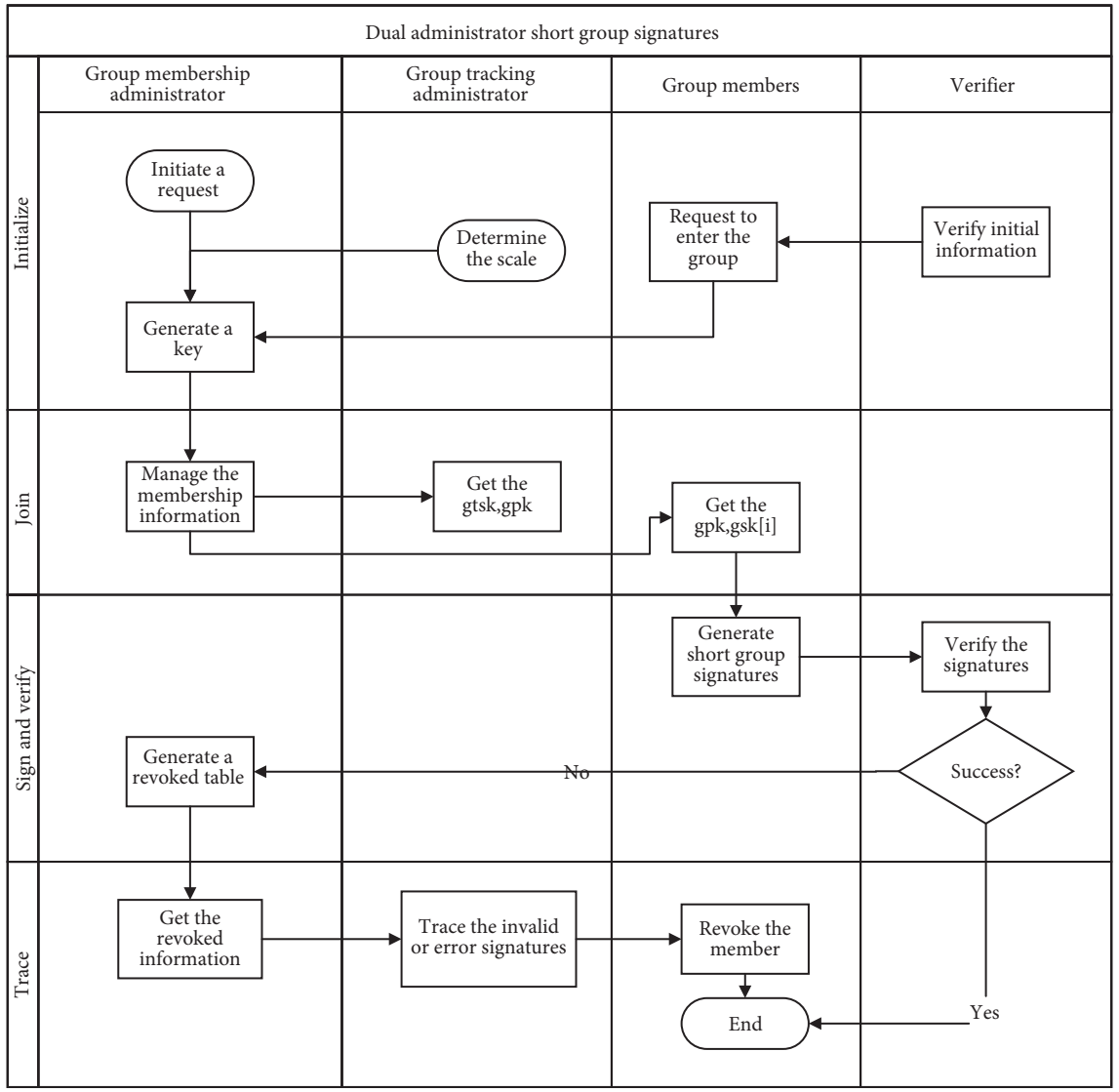

Figure 3: The dual administrator short group signatures swimlane flowcharts.

public key gpk, the group member's private key $gsk[i]$, and the message M. The output is the signature σ.

(d) Verify

Verify (gpk, M, σ): The verification algorithm, that is, the procedure used by the verifier to determine whether σ is a valid signature. The inputs are the group public key gpk, the message M, and the group signature σ. The output is the verification result, which is a Boolean type yes or no.

(e) Trace

Trace $(gpk, gtsk, M, \sigma)$: The tracing algorithm, that is, the signatory member can be traced according to the signature. The inputs are the group public key gpk, the tracing private key $gtsk$, the message M, and the signature σ. The output is the information of the parameters in the group member's private key. Finally, the group member is revoked based on the group membership information.

3.1. Dual Administrator Short Group Signatures Security Analysis. In addition to the basic security standards for the group signature, the short group signatures scheme in the article focuses on the following:

(a) Correctness: Verifying a signature is a process of verifying that a data record is correct. A short group signature σ is a data record in the SDH hypothetical protocol. In this article, the signature σ generated by the short group signatures scheme must be verified by the Verify algorithm.

(b) Anonymity: The verifier of short group signatures verifies the signature using the group public key, so it is impossible to determine which group member

signed the signature, thereby ensuring the group members' anonymity.

(c) Traceability: The group tracking administrator has the key to open the signature at any time to obtain the identity of the group members in the event of a verification failure.

(d) Irreplaceability: Each group member has its own tuple (A_i, x_i, y_i), with the exception of a few public parameters, and the value of y_i can be kept secret by the group members. As a result, no member of the group can generate a signature on behalf of other members, including the group administrator.

4. A Practical Byzantine Fault-Tolerant Consensus Algorithm Based on Dual Administrator Short Group Signatures

There are some issues in the PBFT algorithm, such as high communication overhead, low security, poor scalability, and traceability. To address these issues, the GPBFT algorithm, a practical Byzantine fault-tolerant consensus algorithm based on dual administrator short group signatures, is proposed in this article. The consensus algorithm mainly includes five phases: Request, Pre-prepare, Prepare, GroupSign, and Trace. Firstly, the client initiates a request to the master node in the Request phase. The request is then processed in the Pre-prepare and Prepare phases. The dual administrator short group signatures scheme is proposed in the GroupSign phase of the algorithm, which chooses the group membership administrator as the algorithm's master node. It can reduce the possibility that the master node is a Byzantine node and speed up the view changing and "three-phase consensus" process. Finally, with the involvement of the supervisor, the group tracking administrator can track the identity of the signer by obtaining the signer certificate signer's certificate and the signature information in the Trace phase. The flow chart is shown in Figure 4.

The flow of the GPBFT consensus algorithm is as follows:

4.1. Preparation. In the GPBFT algorithm, the master node N_0, that is, the group membership administrator, should be the first administrator in the short group signatures, who is responsible for the joining and revocation of nodes, and the second node N_1, that is, the group tracking administrator, is responsible for tracking malicious nodes. The selection is based on certification from a reputable CA organization. Therefore, the probability of Byzantine error in the master node is low, which greatly avoids the number of view switches and reduces the cost and communication overhead. In addition, client c acts as the verifier of the short group signatures.

4.2. Request Phase. The client c sends a request $\langle \text{Request}[M, d(M)], o, t, cli\rangle$ to the master node N_0, where M is the message content of the request entity, $d(M)$ is the digest of the message M, o is the operation requested by the client, t is the timestamp, and cli is the client's identifier.

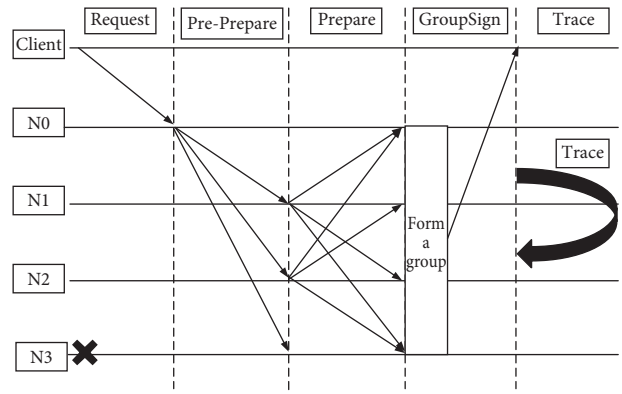

Figure 4: The GPBFT consensus algorithm.

4.3. Pre-prepare Phase. When the master node N_0 receives a request from a client, the message serial number n is first added to the message. Then, the message digest is obtained and signed to generate the signature M. The information is then spliced together and broadcast to the remaining replica nodes.

4.4. Prepare Phase. After receiving the Pre-prepare message from the master node, each replica node verifies the message digest d, message sequence number n, and message signature M in the Pre-prepare message. d must match the message digest of M in the Pre-prepare, the message sequence number n must be in the same view v, and the number is also n. If any of them fail, the Prepare broadcast will be rejected.

The illegal request is discarded. If the request is correct, the replica node i signs the message with its own private key and then sends a Prepare message to other nodes, including the master node.

The entire consensus process must be under the same view in the Pre-prepare and Prepare phases. The non-malicious nodes perform a consistent ordering of the message M, denoted by $\text{Order}(v, M, n)$, where v is the view ordinal number, M is the message, and n is an ordinal number that is confirmed for the message M when View = v.

4.5. GroupSign Phase. The messages in the preparation phase are verified by the master and replica nodes when the Prepare message is received. It mainly verifies whether the messages received by each node are in the same view v, whether the message digest d and the message sequence number n are both consistent, and whether the Prepare message of the replica node is correct. The illegal message requests are discarded after verification, and the legal request messages are carried out in the next steps. The group administrator initiates the request to build a group, and each group member joins to form a group, accepts the message from the Prepare phase, and performs the short group signatures operation. The message serial number remains n. Because of the reauthentication of the group signature, the likelihood of view change is greatly reduced. Even if view change does occur, node revocation can be performed reliably.

The group membership administrator N_0 and group tracking administrator N_1 will be combined with other replica nodes to form a group and generate a short group signature σ through the short group signatures scheme given in the article. Here, all group members can send messages on behalf of the group, and any member of the group who is the first to receive and verify the request message can immediately send the contents of the GroupSign message to the client for verification. The message content is $\langle \text{GroupSign}, d, gpk, \sigma, M \rangle$, where gpk is the group public key and σ is the message signatures generated for the member. After signing with a short group signature, the message sequence number remains n. Because of the revalidation of the new signatures scheme, node revocation can be performed stably even if view change occurs.

When a GroupSign message is received from a group member, the verifier firstly verifies the message digest d and message content M. Then, in the verification algorithm of short group signatures, the message content M, the group public key gpk, and the message signature σ are used as input parameters to verify whether the signature information is correct. If the message is correct, the corresponding information is sent to the group administrator to complete the basic consensus phase. If the message verification is incorrect, the message $\langle \text{GroupSign}, d, M, \sigma \rangle$ is sent back to the group membership administrator for information management updating, and the group tracking administrator checks it and enters the Trace phase.

4.6. Trace Phase. The group tracking administrator checks whether the verified messages M and d are correct and whether σ is a valid signature on M after receiving the error information feedback from the verified client. Then, the group members are tracked according to the feedback information. The Trace algorithm takes the group tracking private key $gtsk$, the signature σ in the feedback information, and the group public key gpk as input and returns the identity of node i in the consensus stage, as well as the identity and information of abnormal nodes.

5. Experiment

To evaluate the performance of the GPBFT algorithm, the Go language is used to simulate the flow of the GPBFT and PBFT algorithm. The experimental environment is an AMD Ryzen 7 4800 h with a Radeon Graphics CPU, 16 GB memory, and 6 GB video memory. The operating system is Ubuntu 64 bit, and the go language version is GO1.15.6. The content of consensus transmission information is set to 48 bytes and 384 bits. The experimental results were processed by Python.

To demonstrate the accuracy and reliability of the GPBFT algorithm, this article compares the GPBFT and PBFT algorithm from five aspects: communication overhead, consensus delay, tracking efficiency, a signature generation time changes with the number of nodes, and basic algorithm security.

5.1. Communication Overhead Analysis. Pre-prepare, Prepare, and Commit are the three main phases of the PBFT algorithm. Because the communication times of the three phases are, $n-1$, $n*(n-1)$, and $n*(n-1)$, the total communication times are $(n-1)+(n^2-n)+(n^2-n)$ or $2n^2-n-1$, and the algorithm complexity is $O(n^2)$.

The Pre-prepare, Prepare, and GroupSign are the main phases of the GPBFT algorithm. In the Pre-prepare phase, the master node broadcasts the Pre-prepare message to other nodes, so communication times are $n-1$. In the Prepare phase, after each node agrees to the request, it broadcasts the Prepare message to other nodes, with the communication times of $n*(n-1)$, that is, n^2-n. In the GroupSign phase, after the group administrator and other nodes form a group, only one node that received the Prepare message needs to sign the short group signatures and then respond to the client. The total communication times in the GroupSign phase are n. Therefore, the communication times are $(n-1)+(n^2-n)+n$, that is, n^2+n-1, and the algorithm complexity is also $O(n^2)$. The communication overhead between them is shown in Figure 5.

5.2. Consensus Delay. Consensus delay refers to the time difference between the initiation and completion of a request. It is an important indicator of the speed of the consensus algorithm. A lower consensus delay allows requests to be confirmed more quickly and makes blockchains more secure and practical. The consensus delay for the test is the time it takes to complete a consensus process, as defined in the following formula:

$$\text{DelayTime} = T_{\text{complete}} - T_{\text{submit}}, \tag{1}$$

where T_{complete} is the time when the client confirmation is completed and T_{submit} is the time when the request starts.

The consensus delay of the PBFT and GPBFT algorithm is investigated by using 4, 25, 50, 75, and 100 nodes, respectively. The results of each group of experiments are the average of 30 different experiments. The consensus delay in the PBFT algorithm includes the time of the Request, Pre-prepare, Prepare, Commit, and Reply phases, plus the time taken by RSA to generate a signature for each node. The consensus delay in the GPBFT algorithm is the total time of the Request, Pre-prepare, Prepare, GroupSign, Trace phases, plus the generation time of dual administrator short group signatures. Because of the characteristics of short group signatures, any member node in the GPBFT group can sign the message anonymously on behalf of the entire group and then feed back to the client. The total time delay of this algorithm should consider the time difference between the first node sending the message and the feedback. If the client verifies correctly, the consensus is complete. If the verification is incorrect, the Trace tracing phase starts. The group tracking administrator opens its group signature and announces the malicious node, and the group membership administrator realizes the cancellation of the member. The consensus time delay between GPBFT and PBFT is shown in Figure 6.

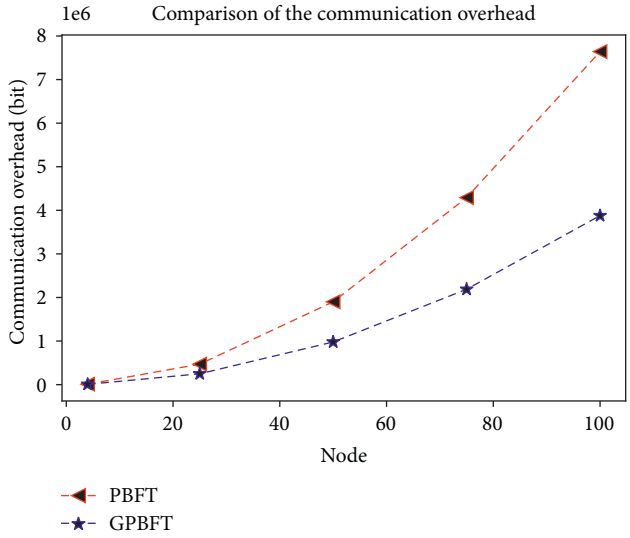

Figure 5: Comparison of communication overhead between the PBFT and the GPBFT algorithm.

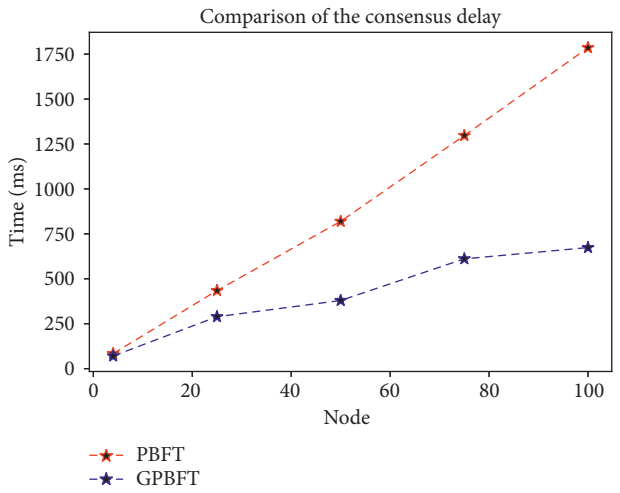

Figure 6: Comparison of the PBFT and GPBFT consensus delay.

5.3. Algorithm Tracking Efficiency. We test the time to trace the group signature taken by GPBFT when the number of nodes is 4, 15, 25, 35, 45, 55, 65, 75, 85, 95, and 105. The experimental results are the average of 50 experiments for each group, in which the abnormal data are excluded. The experimental results show that the tracking time of most nodes is between 1.5 ms and 2.5 ms when the number of nodes is small. The tracking time becomes longer as the node grows, but a few of them remain between 1.5 ms and 2.5 ms. When the number of nodes reaches 75, more than one-third of the nodes' tracking time increases to 70–85 milliseconds; when the number of nodes reaches more than 100, at least half of the nodes' tracking time exceeds 100 milliseconds. The group tracking administrator tracking efficiency is shown in Figure 7.

5.4. The Signature Generation Time of Different Numbers of Nodes in the GPBFT Algorithm. With the growth of nodes, the time for the signature algorithm to generate public and private keys for nodes in GPBFT will also change. We compare the key generation time with different signature schemes between the GPBFT and the PBFT algorithm. This shows the superiority of the short group signatures algorithm in this scheme.

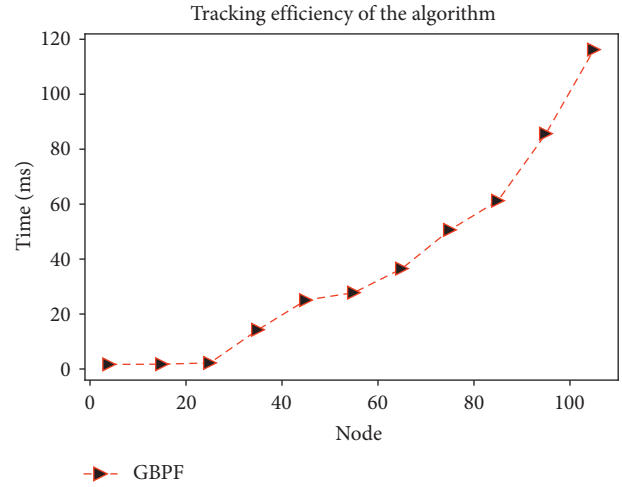

Figure 7: Tracking efficiency of the GPBFT algorithm.

The experimental results show that the GPBFT's short group signatures scheme has the advantage of generation time. When the number of nodes reaches more than 75, this advantage becomes apparent. The results are shown in Figure 8.

5.5. Algorithm Security. Regarding security issues, the PDI framework proposed in Ref. [23] reviews blockchain security research from three aspects: the process level, the data level, and the infrastructure level. Starting from the data level, this article takes a consensus algorithm, authentication, signature scheme, and other aspects as a breakthrough to solve the security problem of blockchain. Data-level security is flanked by process and infrastructure, so the optimized algorithm in this article can also bring beneficial changes to the other two levels. For the consensus algorithm, the following aspects are considered.

5.5.1. Number of Malicious Nodes. As the number of malicious nodes in the simulated network changes, we test whether consensus is reached in the GPBFT algorithm. If there is a single malicious node in the GPBFT, consensus can be reached. If the number of malicious nodes reaches the maximum limit of malicious nodes, consensus cannot be reached. In GPBFT, the Pre-prepare and Prepare phases must still satisfy the Byzantine rules; that is, at least $2f + 1$ messages must be received. The experiments show that the fault tolerance rate of the GPBFT algorithm is consistent with that of the PBFT algorithm. It has normal fault tolerance.

5.5.2. The Master Node Problem. In the PBFT algorithm, the master node is generated by random selection, which has a high level of uncertainty. In contrast, in the GPBFT

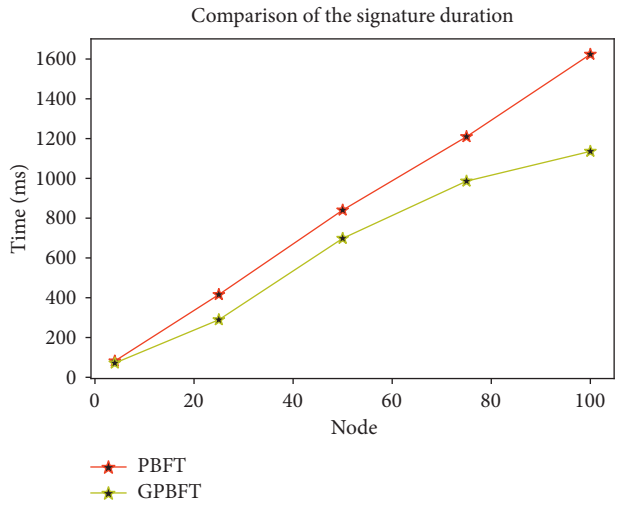

Figure 8: Temporal comparison between the GPBFT's short group signatures and RSA signature.

algorithm, it (i.e., group membership administrator) is generated by the CA authentication institution, and the likelihood of Byzantine error is greatly reduced.

When the master node becomes a malicious node or fails in the GPBFT algorithm, it can be effectively identified and responded to. If the malicious request is propagated by the master node, the consensus system will fail to recognize the request information and will be unable to reach the consensus. As a result, the consensus system will send a view switching request to revoke the malicious master node and then reselect the master node (i.e., the group administrator). Furthermore, when the master node is down, the consensus cannot be reached in the GPBFT.

The tests show that it has good resistance and response ability for the master node problem.

5.5.3. Sybil Attack. In a P2P network without trusted node identity authentication institutions, it is difficult to guarantee that multiple backup nodes are different entities. By deploying only one entity that broadcasts multiple identity IDs to the network, an attacker can act as multiple distinct nodes, and these forged identities are commonly referred to as Sybil nodes. Because there is no God perspective in an entirely decentralized system, no single node will naturally know the exact number of nodes involved. They can only judge the overall situation by the data they received. As a result of this property, the attacking node disguises itself as multiple nodes and broadcasts in the P2P network. The number of Byzantine nodes that can be resisted in PBFT is $N \geq 3f + 1$ (f is the number of malicious nodes).

A dual administrator short group signatures mechanism is introduced in the GPBFT algorithm, in which each node only needs to sign by itself and then feedback the message when it enters the GroupSign phase. Although each node is a group member, each node is relatively independent, and the consensus is reached when the client verifier receives a group member's message. If the attacker who used the Sybil Attack disguises the node, the node will be tracked and revoked.

The consensus can be reached if there is a disguised malicious node in the consensus node. However, the consensus cannot be reached if the total number of nodes remains unchanged and the number of disguised nodes exceeds the maximum number of malicious nodes that the system can bear. When the attacker impersonates the master node and spreads malicious requests, the system will not be able to complete the consensus. At the same time, the system will change its view to overthrow the master node with malicious behavior and then reselect a new master node.

5.5.4. Fault Tolerance Analysis. The maximum number of fault-tolerant nodes of the PBFT algorithm is $f_1 \leq N - 1/3$.

For the GPBFT algorithm, in addition to supporting fault nodes, it also needs to support error nodes. Assume the number of cluster nodes is N and the problematic node is f. Among the problematic nodes, it can be either fault or error or fault and error. There are two extremes.

In the first case, f problematic nodes are both fault and error. According to the features of group signature, at least one node completes consensus in the GroupSign phase. Then, the cluster can reach consensus. This means that the maximum number of fault-tolerant nodes is $(N - 1)$ in this case.

In the second case, the fault nodes and the error nodes are both different nodes. Hence, there will be f fault nodes and f error nodes. When a node is found to be a fault node, it will be excluded by the cluster, and f error nodes remain. Then, according to the features of the group signature, the number of normal nodes in the cluster is at least one. Therefore, there is one correct node, f fault nodes, and f error nodes in all nodes, that is, $2f + 1 = N$. Therefore, the maximum number of fault-tolerant nodes is $f_2 \leq N - 1/2$ in this case.

Due to $f_1 < f_2$, the GPBFT algorithm in this article has a higher fault tolerance than the PBFT algorithm.

5.6. Comparison with Other Literature Studies. For the existing consensus algorithm optimization scheme, we can evaluate it from four dimensions in the actual design, namely, decentralization, efficiency, security, and fault tolerance. The idea of optimization is generally divided into the following aspects: optimizing the consensus process, selecting the primary node, and selecting the appropriate signature algorithm or the underlying communication mode

Buchman [11] replaced messages changing in the PBFT view with variables and deleted the garbage collection mechanism. The simplified Tendermint algorithm only has three stages, which is more concise and understandable than PBFT.

dBFT [24] and Tendermint select nodes based on PoS. dBFT is an algorithm proposed by the AntChain (Neo), which combines PoS with the PBFT mechanism. Although it can improve performance, the election process is static. Because the electoral scheme and results are entirely determined by the project side, the NEO has also been overcentralized. RBFT uses the hash algorithm to group

nodes firstly, and the Raft mechanism is used to elect leaders among groups. Then, the leaders are assembled to run the PBFT algorithm [17].

RBFT, SBFT [25], and HotStuff [13] reduce the system communication complexity to $O(n)$ by introducing threshold signature. Jalalzai et al. proposed the Fast–HotStuff algorithm by using aggregate signature in the NewView phase of consensus, which improves the efficiency of consensus.

The GBC consensus algorithm [26], which is based on the Gossip protocol, improves the fault tolerance of the system from 1/3 to 1/2.

Compared with the above literature, the short group signatures are used as the underlying signature algorithm in this article, which can maintain a certain anonymity, track malicious members in malicious groups quickly and effectively, and increase the stability and security of the algorithm. On the other hand, GPBFT simplifies the algorithm process and reduces the communication overhead of the algorithm, and the introduced GroupSign phase can enter the tracking phase when the consensus fails. The certification authority is used to select the group master node in the article, which reduces the probability of Byzantine errors on the nodes and improves the scalability of the algorithm. In addition, the algorithm is more secure and stable against Sybil attacks with fault tolerance $N - 1/2$.

6. Conclusions

In this article, we proposed the PBFT consensus algorithm (GPBFT) based on dual administrator short group signatures, which combines the advantages of short group signatures with short length, high applicability, low algorithm complexity, and traceable nodes for administrators. Experimental results show that it can not only reduce communication overhead, greatly reduce network traffic, and improve communication efficiency but also be applied in consensus schemes with more nodes, higher scalability, and lower consensus delay. The tracing and dual administrator mechanism in the GPBFT algorithm can make the algorithm more secure and have a greater control rate on malicious nodes. The GPBFT is a weakly centralized algorithm that can be used in the practical applications, including e-commerce, e-banking, e-voting, and e-auction. The selection of the master node and group administrators is an important focus for future work, as it will make the selected administrators more suitable and authoritative and make them more applicable to the alliance chain. In addition, with the rapid development of quantum computing, current blockchain platforms that rely on group signature and hash algorithms are vulnerable to quantum attacks. We can use multichain synchronization and optimized group signature to solve this problem, such as side chain technology or group signatures schemes on lattices [23].

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors thank the Big Data team of the Computer and Information Institute of China Three Gorges University for providing the environment required for the experiment. This research was supported by the Hubei Science and Technology Major Project (Grant no. 2020AEA012) and the National Key Research and Development Program of China (Grant no. 2016YFC0802500).

References

[1] J. Leng, G. Ruan, P. Jiang et al., "Blockchain-empowered sustainable manufacturing and product lifecycle management in industry 4.0: a survey," *Renewable and Sustainable Energy Reviews*, vol. 132, Article ID 110112, 2020.

[2] J. Leng, M. Zhou, J. L. Zhao, Y. Huang, and Y. Bian, "Blockchain security: a survey of techniques and research directions," *IEEE Transactions on Services Computing*, vol. 11, 2021.

[3] Y. Yuan, X. C. Ni, and S. Zeng, "Blockchain consensus algorithms: the state of the art and future trends," *Acta Automatica Sinica*, vol. 44, no. 11, pp. 2011–2022, 2018.

[4] S. M. H. Bamakan, A. Motavali, and A. Babaei Bondarti, "A survey of blockchain consensus algorithms performance evaluation criteria," *Expert Systems with Applications*, vol. 154, Article ID 113385, 2020.

[5] X. Cai, Y. Deng, L. Zhang, and jiuchen Shi, "Blockchain principle and its core technology," *Journal of Computer Science*, vol. 44, no. 1, pp. 84–131, 2021.

[6] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[7] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proceedings of the Appears in the Proceedings of the Third Symposium on Operating Systems Design and Implementation*, New Orleans, LA, USA, February 1999.

[8] X. Li, P. Jiang, T. Chen, X. Luo, and Q Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, vol. 107, pp. 841–853, 2020.

[9] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, and K. Christidis, "Hyperledger fabric:a distributed operating system for permissioned blockchains," in *Proceedings of the 13th EuroSys Conference*, pp. 1–15, ACM Press, New York, NY, USA, April 2018.

[10] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications*, pp. 1–8, IEEE Press, Cambridge, MA, USA, November 2018.

[11] E. Buchman, *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains*, University of Guelph, Guelph Canada, 2016.

[12] F. Saleh, "Blockchain without waste:proof-of-stake," *Review of Financial Studies*, vol. 34, no. 3, pp. 1156–1190, 2021.

[13] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT consensus in the lens of blockchain," 2018, https://arxiv.org/abs/1803.05069.

[14] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, Toronto, Canada, July 2019.

[15] Y. Desmedt and Y. Frankel, "Shared generation of authenticators and signatures," in *Proceedings of the Advances in Cryptology — CRYPTO '91*, Springer, Santa Barbara, CA, USA, August 1991.

[16] M. Du, Q. Chen, and X. Ma, "MBFT: a new consensus algorithm for Consortium blockchain," *IEEE Access*, vol. 8, Article ID 87665, 2020.

[17] D. Huang, Li Lang, B. Chen, and Bo Wang, "Rbft: Byzantine fault tolerant consensus mechanism based on raft cluster," *Journal of Communications*, vol. 42, no. 3, pp. 209–219, 2021.

[18] D. Chaum and E. van Heyst, "Group signatures," in *Proceedings of Eurocrypt of LNCS*, D. W. Davies, Ed., Vol. 547, Springer, Berlin, Germany, 1991.

[19] J. Camenisch and M. Michels, "A group signature scheme based on an RSA-variant," *Basic Research in Computer Science*, vol. 5, no. 27, 1998.

[20] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Proceedings of Crypto1880 of LNCS*, M. Bellare, Ed., Springer, Berlin, Germany, 2000.

[21] X. G. Cheng, J. Wang, and J. X. Du, "Survey on group signature," *Application Research of Computers*, vol. 30, no. 10, pp. 2881–2886, 2013.

[22] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," *An extended abstract of this paper is to appear in Advances in Cryptology—CRYPTO 2004*, Springer, Berlin, Germany, 2004.

[23] J. Leng, S. Ye, M. Zhou et al., "Blockchain-secured smart manufacturing in industry 4.0: a survey," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 237–252, 2021.

[24] Q. Wang, J. Yu, Z. Peng, V. C. Buli, S. Chen, and Y. Ding, "Security Analysis on dBFT protocol of NEO," in *Proceedings of the Financial Cryptography and Data Security*, Kota Kinabalu, Malaysia, February, 2020.

[25] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, and B. Pinkas, "SBFT: a scalable and decentralized trust infrastructure," in *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Portland, OR, USA, June 2018.

[26] Q. W. Zhang, Z. Q. Wang, and Y. Q. Zhang, "Research on trust collection consensus algorithm based on Gossip protocol," *Computer Science*, vol. 47, no. S1, pp. 391–394, 2020.