A Byzantine consensus based on proof-of-work of nodes’ behaviors

Wang Dingyu1,a, Wang Dingmin2,b, Qin Lupin3,c, Xu MinLong4,d

1Xidian University, Xi An, China
2NanChang University, Nan Chang, China 3Nanchang University, Nanchang, China
4Xidian University, Xi An, China
5e-mail:821145448@qq.com, e-mail: 17508928692@163.com,
dloongxu@ourlook.cim

Abstract. Blockchain is a decentralized distributed peer-to-peer system, and decision-making in the system relies on a reliable consensus protocol. Under the current consensus protocol in the blockchain system, the behavior of nodes are not regulated. Based on the existing consensus protocol, this paper analyzes the problems encountered in application scenarios, introduces a credit calculation method to regulate the behavior of system nodes, and with a simplified consistency protocol, it can promote the alliance chain system to enter a virtuous cycle, and improve the efficiency of consensus in scenarios with many wrong nodes.

1. Introduction

The blockchain is, in essence, a decentralized shared database and a decentralized distributed system built based on the chain data structure, distributed node consensus algorithm, cryptography and other technologies[1]. To solve the problem of Byzantine generals in the distributed environment, the consensus algorithm is introduced[2]. The Byzantine problem is an abstraction of point-to-point communication problems, that is, in P2P networks[3], unpredictable problems such as hardware failure, packet dropouts, and even malicious attacks may occur in system nodes, resulting in the failure of multiple proposals to reach an agreement. Under the unreliable channel of message loss, it is important to establish a consistent scheme in the blockchain.

Studies on consistency algorithms have a long history. As early as in the 1980s, Pease and Lamport et al. developed the most famous Byzantine fault tolerant (BFC) algorithm, which relies on the transmission of information between nodes to reach agreement on a protocol. However, due to the complexity of inter-node communication in distributed environment, it was only after the advent of Bitcoin and blockchain technology that BFC came back into the public view. Despite its many flaws, the BFC algorithm offers a new approach to the Byzantine problem, and unlike the proof-of-work (PoW) mechanism, it does not sacrifice computing power to reach consensus.

Castro and Liskov proposed a simple and efficient polynomial Byzantine fault tolerance (pBFT) scheme in the late 1990s, which was the first state-machine replication algorithm applied to the Byzantine problem. Each copy of the state machine can save the state of the service and satisfy the user’s legitimate requests. The system can tolerate at most (n−1)/3 malicious nodes (n is the total number of nodes).
of nodes). However, the pBFT consensus is a protocol that requires a node to act as a leader node, which can only guarantee the system activity in a poorly synchronized network environment. In a completely asynchronous network, the leader node will delay intentionally or unintentionally, disrupting the system’s normal operation.

Since the emergence of pBFT, many studies have been devoted to improving its performance. Among them, CheapBFT[5], FastBFT[6], and MinBFT[4] create a feasible communication environment through secure hardware to reduce the number of participating nodes or communication rounds. CheapBFT, FastBFT, SBFT[7], and ReBFT[8] define a threshold for system error nodes, and if this threshold is exceeded, the protocol will fall back to the "standard" BFT algorithm. However, in the context of large-scale license chain, frequent regression to the "standard" BFT algorithm will degrade the system’s performance and lead the system to an unstable state. The pBFT algorithm based on the idea of committee[9-10] selects network node subset as a "committee" where the participants the small-scale PBFT algorithm to improve the consensus efficiency. However, such reduction in the number of participating consensus nodes increases the degree of centralization and deviates from the original intention of decentralization. The pBFT algorithm[11] based on the idea of short-cycle life signature reduces the communication and computing overhead of the system by using low/medium-security-level (56 or 80 bits) signatures instead of high-security-level (128 bits) signatures, and periodically updates the short-length keys to ensure the security of communication. This scheme is a more practical one. In the alliance chain scenario, the alliance is composed of several organizations in the WAN, and as many nodes as possible are maintained within the organization. The layered pBFT algorithm can be used to localize communication to reduce the WAN communication and thus reduce the system communication overhead. However, this approach delivers good performance only if each organization has more nodes deployed within it.

We note that none of the above schemes has a mechanism to regulate the behavior of nodes in the system and no rules to constrain the behavior of malicious nodes. For the application scenario of alliance chain, this paper introduces a credit calculation mechanism based on the pBFT consensus mechanism. At the same time, some thresholds are set to distinguish the attributes in these participating consensus nodes and give them different permissions and behavior specifications. Also, the credit score is used as the PoW of node behavior to optimize the system and ensure a virtuous cycle.

2. Relevant studies

Consensus algorithms, as a core technology of building trust in blockchains, enable distrusting actors to verify and agree on the validity of transactions in a blockchain in a decentralized manner. The efficiency of consensus algorithms directly affects the overall performance of a blockchain system. Ensuring consistency of distributed systems efficiently under the Byzantine problem is important to improve the systems’ performance of distributed system. The decision-making power of the system is distributed among all participating nodes. A larger the size of the members, thenodes often means a higher the calculation and communication overhead for the consensus. If a distributed system suffers only from passive errors, such as packet dropout and delay, consistency can be solved through Raft and Paxos algorithms[12]. If the participating consensus nodes in the system do not know each other and are motivated by certain interests, a Byzantine problem may occur where some nodes actively send error messages to other nodes. In this system, a consistent algorithm with fault tolerance is needed.

2.1. Proof-of-work

Satoshi Nakamoto in his paper[13] proposed the first PoW consensus algorithm applied to block chains. In the PoW consensus mechanism, each node needs to solve a cryptography problem that is difficult to verify. The node that solves the first problem has the right to generate blocks and receive a profit. PoW is not a new concept, but a practical distributed consensus system formed by workload proof algorithm and Merkle Tree, digital signature and P2P network, etc. PoW consensus contributes to the invention and development of Bitcoin. When Bitcoin was invented, the PoW consensus algorithm well guaranteed the security and fairness of Bitcoin. With the development of blockchain, however, PoW reveals some of its
obvious weaknesses. The waste of resources caused by the immense computing power required by PoW algorithm is a major problem. Secondly, the block interval is too long, resulting in low efficiency in transaction confirmation. When a special chip appears, a few institutions with strong computing power monopolize the block productivity, thus reducing its fairness, the system’s accounting accuracy and its resistance against attacks.

2.2. Equity certificate
In order to solve the monopoly and waste of PoW computing power resources, a consensus solution of proof of stake (PoS)[14] was proposed. In the system, nodes with a higher computing power do not have higher accounting interests. At the beginning of the PoS algorithm, the mining method in PoW is adopted to realize fair distribution of tokens among the miners. In the later stage, PoS will take over the algorithm maintenance of the whole system as the mining difficulty increases, and a new concept “coin days” is introduced. When miners dig a mine, they use the proof of interest instead of the proof of work. Compared with PoW algorithms, PoS algorithms reduce the waste of computing power and the transaction confirmation time. Therefore, PoS gained increasing popularity for digital currencies after the advent of Bitcoin. However, PoS algorithms have some nodes controlled by core interests, which to some extent violates the original intention of decentralized distributed ledger.

2.3. DPoS(delegated proof-of-stake)
DPoS(delegated proof-of-Stake)[15] adopts a representing mechanism that weakens the degree of decentralization to some extent. That is, all participating nodes vote for part of the nodes that have the right to witness generation of blocks, i.e. witness nodes. In this way, the number of participating consensus nodes can be reduced and the time to reach a consensus can be reduced. When the witness node’s ability to synchronize the new block is weakened, the transaction information cannot be correctly logged, and the new block synchronizes in a timely manner. Other participants can vote to replace it. The revenue generated by the block will encourage the selected witness nodes to earnestly fulfill their responsibilities and safeguard their rights during the consensus process.

2.4. Byzantine fault-tolerant algorithm
The purpose of pBFT algorithm[16] is to ensure the consistency and correctness of final decisions in an unreliable network environment. The algorithm can guarantee the availability and security of the system when the total number of unreliable, or malicious, nodes is less than one third of the total. In a pBFT algorithm, all nodes are divided into three types: clients, the master node and Replica nodes. The process is divided into a consistency protocol, a view-change protocol and a checkpoint protocol. During block generation, nodes run the consistency protocol, obtain other states through point-to-point communication, decide their own states, and complete the unification of block chain data of each node through a process similar to voting. In case of timeout failure of the consistency protocol, the system replaces fault nodes with a view-change protocol to ensure stable operation of the system. The checkpoint protocol periodically removes expired interactive data to reduce the node storage burden, and is responsible for periodically checking whether the system state is unified and synchronizing inconsistent nodes. There is a large amount of point-to-point communication between pBFT nodes, and consensus is reached through negotiation, which takes up a large amount of communication resources. Therefore, it is of great significance to optimize the communication process and improve efficiency to reach a consensus.

3. RPBFT protocol model
3.1. Network environment model
This paper assumes that the alliance chain system model is partially synchronous in the operating environment, that is, the sending delay has an upper limit. The whole network is accessible, the transmission of messages between honest nodes must be guaranteed to the destination address, and the
transmission channel must not lose, copy, or modify messages. The network is dynamic, that is, the whole network can sense the entry and exit of nodes, which can be assumed by configuring a trusted certificate authority in the alliance chain system.

3.2. Adversary capability model
In this paper, it is assumed that there are altogether \( n = 3f + 1 \) nodes in the federated chain system, and \( F \) is the maximum number of adversarial nodes at any time in the system. The behavior of the adversarial nodes can be arbitrary, such as colluding to send error messages or staying silent. The presence of an adversarial node can delay, lose, duplicate the message moving through the network, but this article assumes that an adversary cannot delay the right node indefinitely. In addition, it is also assumed that the attacker's computing power is limited, and the password algorithm used by the alliance chain system cannot be cracked, so that the original information cannot be obtained by the reverse operation of the digital abstract, the signature of the correct node cannot be forged to carry out impersonation attacks, and two messages with the same abstract cannot be found.

3.3. Calculation mechanism of credit score
In order to effectively regulate the behavior of nodes in the system, this paper introduces a credit calculation mechanism. On the one hand, the credit score can be used to dynamically adjust the nodes’ credit and give them different attributes and permissions. On the other hand, it can be used as a reward to motivate nodes to follow the rules. This paper gives the following definitions for credit calculation:

**Definition 1** A credit score is a reference index to measure a node’s degree of credibility. The score value \( R \) is a real number between 0.1 and 1.0. The higher the score, the higher the credibility of the corresponding node. For nodes newly added to the alliance chain system, their score value is initialized to 0.5.

**Definition 2** Credit score rewards and punishments will be given based on dynamic changes in node behavior during the consensus process. Let \( R_V \) represent the score of a node in View \( V \), then the relationship between the score of this node in view \( V+1 \) and \( R_V \) is as follows:

- If the master node generates a valid block, or if the transaction message of the slave node is the same as the final result, its section rating will gradually increase. The calculation formula is as follows:
  \[
  R_{V+1} = R_V + a|\lg(R_V)|(0 < a \leq 0.1)
  \]
  The purpose of using logarithms is to limit growth rates as reliable node creditworthiness increases. This increases the difficulty of obtaining a high credit score, whereas parameter \( A \) is an adjustment that needs to be made in a real scenario.

- If the master node fails to generate a new block, or if the slave node message is inconsistent with the final result, or if it remains silent, then the corresponding credit value will decline linearly. Its formula is:
  \[
  R_{V+1} = bR_V (0 < b < 1)
  \]
  The descending gradient of the credit value depends on the parameter \( b \), which should be set according to the requirements of the specific application scenario.

- If one node sends inconsistent messages to other consensus nodes, that node will be regarded as a Byzantine node and its reputation will be directly reduced to the minimum value, i.e.:
  \[
  R_{V+1} = 0.1
  \]

**Definition 3** The credit level is an important basis that determines the power of a node. The level depends on the size of the node reputation value \( R \). In the system, according to the nodes’ behavior score
values, three threshold ranges I, II, and III are defined, and the nodes are classified into three types, as shown in Table 1.

| credit ratings | Credit score range | Role           |
|----------------|--------------------|----------------|
| I              | $R \in [\beta, 1.0)$ | ✓              |
| II             | $R \in [\alpha, \beta]$ | ✓              |
| III            | $R \in (0.1, \beta]$ | ✓              |

Table 1 Node permissions and thresholds

In the table, $\alpha$ and $\beta$ are two thresholds for the node status changes, the size of which is determined by the running conditions of the system; $0.5 < \alpha < 1.0$ and $0.1 < \beta < 0.5$ means that the reputation value is greater than the $\alpha$ on Level I, while lower than the $\beta$ on Level III.

**Definition 4** in principle, should be given III level node "rehabilitation", then gives the following rules: each $T$ time period, in III level node score value will rise $c$, but not more than $\beta$, the specific calculation formula is:

\[
RV_{t+1} = \min(\beta, RV_t + c(t/T))
\]  

(4)

In Equation (4), $T$ is the total time from the last time the node participated in consensus to now, and $c$ is determined with the actual application scenario.

**Definition 5** To ensure the security of the alliance chain system, the malicious nodes in the system should be removed in a timely manner. According to Equation (4), available reputation value to recover from a minimum of 0.1 time $t$ to the $\beta$ needed for this article defines "long-term" as at $3t$ in $10t$ of time cycle time, and a node with the level of long-term III is identified as a malicious node.

3.4. Consistency Protocol

The pBFT algorithm aims to solve the problem of how to guarantee the consistency and correctness of the final decision even if there are malicious nodes. Blockchain systems that use the pBFT consensus mechanism have $n$ servers, System nodes include normal nodes and nodes with Byzantine errors. Proper operation of the system requires that $n \geq 3F + 1$, and the number of normal nodes is at least twice the number of Byzantine nodes. To ensure the consistency and security of the system, the following requirements should be met:

- The block chain data and transaction information stored by normal nodes are correct and consistent;
- If the block data is confirmed to be correct, the normal node must accept the block.

The nodes need to broadcast and exchange messages, called certificates, to reach a consensus. The certificate includes the number of the certificate, the identity information of the sender, the agreement of the negotiated content, etc. When a node receives the information, it checks the information and records it in the log.

3.4.1. Complete conformance Agreement

In alliance chain systems, a number of transactions or changes in account status are packaged into blocks at regular intervals. The nodes in the system guarantee the correctness and consistency of the record block information by the consistency protocol. There are two types of roles in a protocol: the master and the slaves. There is only one master node in a consensus, which is responsible for validating transactions received, and the validated transactions are packaged into blocks. Nodes participating in the consensus will have different Numbers from 0 to $R-1$. The selection formula for the master node is as follows:

\[
P = (h + v) \mod(|R|)
\]  

(5)

Where $P$ is the number of the selected node; $V$ is a view number that increments from zero; $|R|$ is the total number of nodes participating in the consensus. In general, the pBFT, as shown in Figure 1,
implements a complete consistency protocol, and all nodes need to go through three major phases to reach a consensus: preparation, preparation, and validation.

1) Pre-Preparation: The master node sorts and packages the requests from the client into blocks, generates pre-preparation certificates, issues prepared certificates to the slave nodes, and then enters the preparation stage.

2) Preparation: After receiving the prepared certificate from the master node, if it is the first time a node receives the certificate, the node enters the preparation state and forwards the certificate to other slave nodes.

3) Commit: After collecting 2F +1 matching acknowledgement messages, the node can determine that a consensus has been reached, execute the transaction in the block, and return the corresponding result to the client.

All valid messages received by a node should be added to its local message log throughout the process. If the primary node fails, the slave node will trigger the view-change protocol to select a new primary node and move to the next view, thus ensuring system viability. Checkpoint protocols are executed periodically to ensure consistency in the state of all nodes.

![Figure 1. Complete PBFT conformance protocol](image)

3.4.2. Simplified conformance protocol

The complete consistency protocol needs to complete two highly complex $O(n^2)$ communication processes. Therefore, in order to simplify the communication process, this paper, referring to mixed group Byzantine fault tolerance, divides the implementation consistency agreement area into cases with and without Byzantine fault. The simplified consistency protocol is executed when there is no Byzantine error, and the entire consistency protocol is returned when a Byzantine error occurs. The simplification process is shown in Figure 2:

1) The master node gives the prepared certificate to all the slave nodes. When a slave node receives the certificate, if the master node recognizes the content of the certificate, it will reply with the approval message;

2) If the master node receives three approval messages, it packages the approval information and sends it to all slave nodes. Slave nodes can verify whether the approval information of other nodes is correct. If it is correct, then all nodes receive the block transaction information, all nodes enter the validation state, and execute the block transaction in the block.

3) If the master node does not receive all the approval information, it enters the full consistency protocol flow.
3.5. View-change protocol

A view is the definition of the node relationship in the pBFT consensus mechanism. The number of the view is denoted as $V$, the nodes in the view have different numbers, and each view has a primary node. In the consistency protocol, the master node has the core right to record transactions to the block, and the fault of the master node will cause the block to stop. The view-change protocol completes the task of changing the master node when the master node fails to ensure the operation of the system, and the view number increases by 1 after the change of the view. The view-change protocol is triggered by the slave node and takes the timestamp $R$ of the latest block in the blockchain as the starting time. There are two trigger conditions for timeout in line with the consistency protocol:

- No prepared protocol from a new master node was received within a limited time $\Delta T_1$;
- No new block generation was completed within a limited time $\Delta T_2$ ($\Delta T_1 < \Delta T_2$).

Either of the above two conditions can be considered a primary node failure, and a view change is required. View change requires interactive communication between nodes. The execution process is as follows:

- Execute a view-change protocol from a node to enter the view $V+1$, and send the view-change certificate to all nodes, including the latest block number and summary information, as well as the new view number and master node.
- If a node receives $2f+1$ view-change certificate, it will send the certificate to the main node in the view $V+1$. At the same time, the consistency protocol certificate received after $R$ is cleared, waiting for the new master node to initiate the consistency protocol.
- The master node starts a new round of consistency protocols according to its own stored blockchain data.

The view-change process causes the transaction confirmation to stop, so you need to avoid primary node errors as much as possible. Through credit accumulation, nodes that have successfully completed block generation for many times will have more opportunities to serve as the master node and improve system efficiency.

3.6. Checkpoint protocol

Ideally, all nodes can complete all interactive tasks in the system in a timely manner and maintain consistency. However, as some nodes may fall behind other nodes due to network delay or failure, a periodic protocol is required to synchronize the whole system to prevent system failure caused by inconsistent accumulation of nodes.

In the traditional pBFT algorithm, the replica node collects $2f+1$ checkpoint message as evidence. Due to the high computing overhead of collecting evidence, the evidence is calculated after the Checkpoint_Interval (usually 100 times) requests are executed, and the resulting state can be called Checkpoint.

In this agreement, the checkpoint protocol, after checking the consistency status, clears the certificates related to the recognized blocks to reduce the node storage pressure. In this paper, we also
extend the function of dynamic adding and deleting nodes and credit rating sorting in the checkpoint protocol. The execution time interval of the checkpoint protocol is recorded as CT. Each execution node asks for block chain status information from other nodes. Once it is found inconsistent with most nodes, it will actively ask other nodes for block information starting from the last checkpoint. After the synchronization is complete, the node updates the local transaction list to clear the transactions that have been logged, and to clear the certificates that precede the latest block timestamp. For transactions that have been logged, the certificate before the latest block timestamp is cleared.

4. Protocol analysis & simulation

4.1. Analysis for protocol

4.1.1. Correctness analysis
A perfectly correct blockchain system must satisfy two important characteristics of distributed systems: security and availability. In the alliance chain scenario, safety and activity are defined as follows:

- Security means that transactions in each agreed block are valid and that the honest node is the same block at any given height.
- Availability refers to the fact that the alliance chain system responds correctly to a client-committed transaction in a limited time.

In this paper, it is stipulated in the adversarial power of the cipher algorithm used by the alliance chain system cannot be cracked, so the signature of the correct node cannot be forged to carry out identity impersonation attacks, and the original information cannot be obtained by the reverse operation of the digital abstract. Next, the consistency of blocks in local storage of all honest nodes will be proved, that is, if valid blocks $B_1$ and $B_2$ are respectively submitted locally by two honest nodes in a round of consensus, then $B_1 = B_2$.

Proof: In the alliance chain, if $B_1$ is valid, then at least $2f_k + 1$ nodes recognize transactions in $B_1$ and commit $B_1$ locally. Similarly, $2f_k + 1$ nodes need to acknowledge transactions in $B_2$ and commit $B_2$ locally. Therefore, at least $2(2f_k+1) - (3f_k + 1) = f_k + 1$ nodes simultaneously identify $B_1$ and $B_2$. There are at most $f_k$ error nodes and at least one honest node. The honest node will strictly adhere to the protocol and will not submit two different blocks in a round of consensus, so it must satisfy $B_1 = B_2$.

4.1.2 Analysis for Fault tolerance
In an alliance chain system, the ratio of Byzantine fault nodes is the most important factor affecting the performance of the system. RpBFT does not introduce strict scenario assumptions in fault tolerance and provides the same fault tolerance capability as pBFT. The number of error nodes tolerated is $(n-1)/3$, where $n$ is the total number of nodes.

4.1.3 Communication overhead analysis
The problem with pBFT and its derived consensus mechanism is that the consensus process requires a lot of communication between nodes. In RpBFT, due to the inclusion of checkpoint protocol in credit score calculation, additional communication process is required to reach an agreement on credit score. Data transmission will increase within a short time after the system starts to run or when error nodes in the system are relatively few. However, in the scenario with many error nodes, the invocation of view switching protocol can be reduced, thus reducing the amount of communication data.

4.2. analogue simulation

4.2.1. simulation for Fault-tolerant
In this paper, Matlab multi-thread simulation alliance chain system is used. In the system, a total of 91 consensus nodes are set. Under the random variation of the number of unreliable nodes within 1 to 30, the transactions per second (TPS) generated by pBFT and RpBFT during 0 to 20 min of system
operation are compared on average. As shown in Figure 3, RpBFT and pBFT can achieve the same efficiency in a short time.

![Figure 3. Comparison of transactions generated by RpBFT and pBFT per unit time](image)

4.2.2. Communication overhead simulation
Without considering fault tolerance, this paper still uses Matlab multi-thread simulation alliance chain system to set up 181 consensus nodes and linearly increase the number of error nodes within the range of 1~90. As Figure 4 shows, when the system starts to run, RpBFT generates unit blocks less efficiently than pBFT in order to agree on additional credit scores. Over time, as the number of error nodes in the system increases, the pBFT tends to stabilize and the RpBFT becomes more efficient.

![Figure 4. Comparison of unit block volume generated by PBFT and RPBFT](image)

5. conclusion
The emergence of the pBFT consensus mechanism ensures the consistency and correctness of the final decision under the unreliable network environment, but how to regulate the behavior in nodes is the problem to be solved at present. This paper proposes RpBFT, which consists of a credit value calculation mechanism and a simplified consistency protocol. On the one hand, the credit value is taken as the proof of work of the node’s behavior to make the node comply with the rules. On the other hand, when there is no error, the alliance chain system enters the simplified consistency protocol, which reduces the system communication overhead. Through these two mechanisms, the system enters a virtuous cycle. Finally, simulation results show that RpBFT is superior to pBFT in the case of many error nodes.

The consensus of proof of work for node behaviors proposed in this paper has higher efficiency than pBFT when there are many error nodes, but there are still many deficiencies. How to ensure the
transmission of messages without packet dropouts and occurrence of errors in the large-scale scenario is the direction of future work.

References:
[1] Bonneau J, Miller A, Clark J, et al. SOK: research perspectives and challenges for bitcoin and cryptocurrencies[C]//Proceedings of the 2015 IEEE Symposium on Security and Privacy. Washington, DC : IEEE Computer Society, 2015 : 104-121.
[2] Eyal I, Cencer A E, Sirer E G, et al. Bitcoin NG: a scalable blockchain protocol[C]//Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation. Berkeley, CA : USENIX Association, 2016 : 45—59.
[3] Amalarethinam D I G, Balakrishnan C, Charles A. An improved methodology for fragment reallocation in peer-to-peer distributed databases[C]//Proceedings of the 4th International Conference on Advances in Recent Technologies in Communication and Computing. Piscataway, NJ : IEEE, 2012 : 78-81.
[4] Veronese G S, Correia M, Bessani A N, et al. Efficient Byzantine fault-tolerance [J]. IEEE Transactions on Computers 2011, 62(1): 16-30.
[5] Kapitza R, Behl J, Cachin C, et al. CheapBFT: resource-efficient Byzantine fault tolerance[C]//Proceedings of the 7th ACM European conference on Computer Systems, Bern, Switzerland, April, 2012: 295-308.
[6] Liu J, Li W, Karame G O, et al. Scalable Byzantine consensus via hardware-assisted secret sharing [J]. IEEE Transactions on Computers, 2018, 68(1): 139-151.
[7] Gueta G G, Abraham I, Grossman S, et al. SBFT: a scalable and decentralized trust infrastructure [C]//2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Portland, OR, USA, June, 2019: 568-580.
[8] Distler T, Cachin C, Kapitza R. Resource-efficient Byzantine fault tolerance [J]. IEEE Transactions on Computers, 2015, 65(9): 2807-2819.
[9] Jalalzai M M, Busch C, Richard III G. Proteus: a scalable BFT consensus protocol for blockchain[J/OL].arXiv preprint arXiv, 2019, 1903.04134.
[10] Meng Y, Cao Z, Qu D. A committee-based Byzantine consensus protocol for blockchain[C]//2018 IEEE 9th International Conference on Software Engineering and Service Science (IC-SESS), Beijing, China, November, 2018: 1-6.
[11] Fan X. Scalable practical Byzantine fault tolerance with short-lived signature schemes[C]//Proceedings of the 28th Annual International Conference on Computer Science and Soft-ware Engineering, Markham, Ontario, Canada, October, 2018: 245-256.
[12] Mport L. Paxos made simple[J]. ACM SIGACT News, 2011, 32(4) : 18-25.
[13] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [Z]. Bitcoin White Paper, 2008.
[14] Fazio N, Nicolosi A. Cryptographic accumulators: definitions, constructions and applications[EBOL]. 2002. http://www.cs.nyu.edu/nicolosi/papers/accumulators.Pdf.
[15] Fan Jie, Yi Letian, Shu Jiwu. Research on the technologies Byzantine system[J]. Journal of Software, 2013, 24(6) : 1346-1360.
[16] Yuan Yong, Wang Feiyue. Blockhain :the state of the art and future trends[J]. Acta Automatica Sinica, 2016, 42(4) : 481-494